INTELLIGENT QUALITY MANAGEMENT SYSTEM FOR FLEXIBLE MANUFACTURING SYSTEMS

By

THABO GEORGE BIHI

Thesis submitted in fulfilment of the requirements for the degree:

Doctor of Engineering in Electrical Engineering

In the Department of Electrical, Electronic and Computer Engineering

Faculty of Engineering, Built Environment, and Information Technology

Central University of Technology, Free State

Promoter: Prof. K. Kusakana

August 2022

DECLARATION

I, THABO GEORGE BIHI, student number _____, do hereby declare that this research project, which has been submitted to the Central University of Technology Free State, for the degree: Doctor of Engineering in Electrical Engineering, is my own 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, Free State.

This project has not been submitted before by any person in fulfilment (or partial fulfilment) of the requirements for the attainment of any qualification.

T.G. BIHI Date: 30 August 2022

DEDICATION

I thank God for his blessings, mercy and guidance on this path I've walked, as well as the knowledge and wisdom that I have uncovered along the way. It would be amiss of me not to thank those that came before me, those that created a resilience and strength in me that, at first, I didn't realise was there. "*Ke leboga Modimo le badimo*", as we are prone to say in my native language.

I dedicate this research and achievement to my late parents, Matthews Lekula Bihi and Thandiwe Josephine Bihi, who made me the person I am today, in the time that we were blessed to have them in our lives. I know that you are looking down, with proud smiles on your faces and that is enough to keep me moving forward, always. On this journey I have been fortunate enough to have a supportive, encouraging and loving partner. Thank you for what you did to get me through all the challenging times.

An unequivocal thank you goes to my siblings, as well as my nephew and niece for the love and laughter. I couldn't have wished for a better support system. The conversations we shared pulled me out of some truly sombre moments.

To my friends and colleagues, I am eternally grateful for your input and contributions, big and small. I hope that I have helped you as much as you have helped me on this journey. Your presence and positive attitudes helped in more ways than one, encouraging and prompting me to be more than I had imagined before.

Thank you and God bless everyone that has been with me throughout this progression. You were a blessing I didn't know I needed.

ACKNOWLEDGEMENT

A profound appreciation for my research supervisor, Professor Kanzumba Kusakana, for the support, patience and your availability throughout this period. It is your guidance and belief that carried me to the finish line. It is truly an honour and privilege to work with you.

I would like to acknowledge the support and assistance afforded me by the Central University of Technology, Free State (CUT) and the Department of Higher Education and Training (DHET). Through the DHET driven Staffing South Africa's Universities Framework (SSAUF), I was given an opportunity to be part of the New Generation of Academics Programme (nGAP) and I am truly grateful for that.

Thank you for the generosity.

LIST OF ABBREVIATIONS

AGV	Automated Guided Vehicle
CB	Circuit Breaker
CPSs	Cyber-Physical Systems
DB	Database
FMS	Flexible Manufacturing System
GUI	Graphic User Interface
ID	Identifier
IDE	Integrated Development Environment
IoT	Internet of Things
I/O	Input and Output
IMS	Intelligent Manufacturing System
IP	Internet Protocol
JIT	Just-In-Time
LED	Light Emitting Diode
NC	Numerical Control
QMS	Quality Management System
QR	Quick Response
SQL	Structured Query Language

ABSTRACT

The Industrial Revolution was the transition to new manufacturing processes in Great Britain, Europe and the United States. The transition included going from hand production methods to machines. The result was new chemical manufacturing and iron production processes, the increase in the use of steam power and waterpower, the development of machine tools, as well as the rise of the mechanized factory system. That was the First Revolution which began in the late 1700's in Britain. The Second, involved the use of electric power to create mass production, beginning in the late 1800's, in the United States. The late 1900's gave way to the Third Industrial Revolution. During this Digital Revolution, electronics and information technology were used to automate production. The current, Fourth Industrial Revolution, is building on the Third. Industry 4.0 holds the promise of increased flexibility in manufacturing, along with mass customization, superior quality and improved productivity. Industry 4.0 has evolved from the 'Industry 4.0' vision, an initiative for increasing Germany's manufacturing industry competitiveness. The scope of Industry 4.0 was initially restricted to the digitalization of production processes at the factory level. Technology trends of Industry 4.0, refer to a wide variety of information, digital, operations and advanced manufacturing technologies that, collectively, push the digital industrial revolution.

Intelligent manufacturing is based on the concept of optimizing manufacturing and productivity, by taking advantage of technological advances. Flexible Manufacturing Systems (FMS's) are a form of intelligent manufacturing. A Flexible Manufacturing System reacts to changes in the production process. This includes changes in the product and the production schedule. Quality management processes should be increasingly efficient and effective, which is becoming a competitive factor for companies. Disadvantages of FMS, include its higher upfront costs and more time required to design the system specifications for a variety of future needs. Furthermore, there are costs associated with the need for specialized technicians to run, monitor and maintain the FMS. A traditional Quality Management System (QMS) cannot be used for a Flexible Manufacturing System, as it is unable to keep up with the changes in the FMS.

The aim is to design, construct and evaluate an Intelligent Quality Management System (iQMS), which may learn and adapt to the change of the Flexible Manufacturing System (FMS). The emphasis of the system design is on easy initialisation, quick transition from learning to running and data handling.

An initial review is carried out on the available literature on the Fourth Industrial Revolution, and flexible manufacturing systems. The application of quality management in the manufacturing industry, is investigated. The literature on flexible manufacturing is used to provide context and highlight the problem statement.

A case study is used to showcase the challenges encountered, when circuit breakers of varying specifications are set up in an assembly system with interchangeable tools/stations. This highlights the assumptions that led to the formulation of the project. It provides a concise description of the study area and the methods employed. The tools used in the development of the QMS project, are discussed.

The development of the Quality Management System, is discussed, detailing the hardware and software aspects of the research study. The details on the use of the ARDUINO platform to construct the QMS's Hardware Nodes, are included in this Chapter. Furthermore, QMS Software development and deployment details are incorporated. The innovative use of QR codes to collect training data is highlighted in the Chapter. QR code decoding and training data extraction is exhibited in this section of the study.

The concluding Chapter discusses the results from the iQMSs testing and data collection process, as well as observes future work. The main contributions of the study, in comparison with further available research, regarding the manufacturing industry and industry 4.0, are further highlighted. The study realised a data management system that caters to the unique data management requirements of Flexible Manufacturing Systems. A major highlight of the study shows the benefit of reworking quality management practices using new digital advances.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

1.1. BACKGROUND

The first three industrial revolutions came about as a result of mechanization, electricity and information technology (IT). The introduction of the Internet of Things and Services into the manufacturing environment, is fostering a Fourth Industrial Revolution, where the smart optimization and computerization of all the factors and phases of the manufacturing process, including the conceptualization and design of a product, as well as its production and transaction, take a leading role [1]. This next generation of industry, Industry 4.0, holds the promise of increased flexibility in manufacturing. This, furthermore, means improvement in mass customization, quality, and productivity. Industry 4.0 is an initiative, with the aim of inaugurating intelligent factories, wherever production technologies are upgraded and transformed. This transformation comes with the aid of Cyber-Physical Systems (CPSs), the Internet of Things (IoT) and Cloud Computing. In the new industrial era, manufacturing systems may monitor physical processes and make smart decisions through real-time communication and interaction. Intelligent manufacturing is based on this concept of optimizing manufacturing, by taking advantage of new technological approaches [2] [3]. The manufacturing system is influenced by many different factors, which are 'types of operations', 'number of workstations', 'automation level' and 'system flexibility'. Flexible solutions differ on account on the flexibility they provide. Some focus on organizational aspects are based on implementing management solutions, resulting in flexibility of operating time available, flexibility of competencies and operators' skills (multi-skilled operators), as well as general flexibility of resources benefitting from shared resources. Others, embrace modern technologies to, in the end, obtain similar results: flexibility of time, operation and resources in general. The basic principle of Industry 4.0 is that, by connecting machines, work pieces and systems, businesses are creating intelligent networks along the entire value chain, that can control each other autonomously. This is the reason as to why we additionally name the Industry 4.0 in terms

such as: “factory 4.0” or “smart factory”. This designation is a continuation of the term “digital factory”, in previous years [4].

The manufacturing industry is undergoing revolutionary information and intelligent transformation [5]. Manufacturing companies that have started to apply Industry 4.0 concepts and systems, are found in various industrial sectors. Examples of these sectors include automotive, transportation, aerospace and heavy machinery. These industrial sectors have their unique characteristics and, hence, different journeys within the applications of digital transformation [6].

Flexible Manufacturing Systems (FMS) may facilitate the progress towards enterprise integration. It has been reported in many studies that FMS makes the transition to agility faster and easier. In addition, FMS may impact nearly every process at the operational level. There have been many definitions of the term FMS. These definitions focus on the major elements or components of FMS, such as: flexible machines or workstations, automated material handling systems and computerized networks, that direct and link all the related processes. However, FMS promises change. In FMS, NC machines on the line are controlled by a computer, robots handle the parts and automatically guided carts carry completed products to their following locations. Automatic tool-changing systems are incorporated, and product changeover is included in the computer program [7]. A Flexible Manufacturing System reacts to changes in the production process. This includes changes in the product and the production schedule. These changes present a problem: traditional Quality Management System cannot be used for Flexible Manufacturing Systems because they are unable to keep up with the changes in the FMS [8]. The adoption of modelling and simulation techniques, as well as their implementation in the industrial sector in order to digitize production and business processes, is done differently; there is no unanimously accepted methodology [9].

Organizations and enterprises, increasingly depend on intelligent information systems, that operationalize corporate data and knowledge in a unified way across system boundaries. The difference between such systems and classical automatic systems, is that the task synthesis and decision-making realize within the system [10]. Intelligent manufacturing covers many aspects of the manufacturing field, not solely a technology but, further, the integration of all aspects of manufacturing field and information

technology, aiming to convert data acquired across the product lifecycle into manufacturing intelligence, to yield positive impacts on all aspects of manufacturing [11].

1.2. PROBLEM STATEMENT

The main disadvantage of a FMS in dealing with manufacturing resources, such as time and contribution to produce a new product, is its high flexibility. There are many applications of FMS, however, the most widely applied application is found in the manufacture of small groups of products, largely from mass production [12].

The challenge in flexible manufacturing that led to this study, may be explained with the following case study, that involves a flexible system tasked with the assembly of circuit breakers. Assuming that all tests are carried out on the same line, as seen in Arrangement 1 (Figure 1.1). Thereafter, the arrangement should change for any reason, to either 2 or 3. The need for Intelligent Quality Management System, arises in Arrangement 1 to log the data of the circuit breakers, while at the same time, automatically detecting whether it qualifies as a pass at the 5 amperes, 10 amperes or 20 amperes electrical test. The test defines the type of circuit breaker that is under assembly.

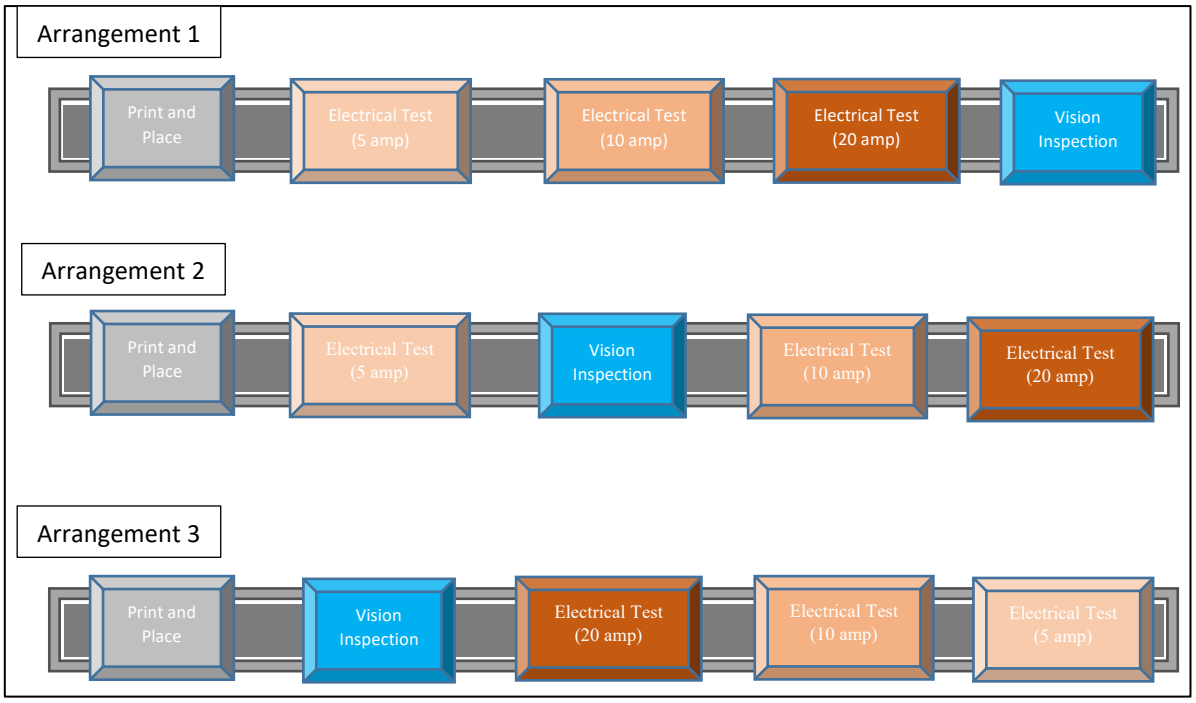


Figure 1.1: Assembly System Configurations

One of the disadvantages of traditional manufacturing facilities, is that a change of the FMS to Arrangement 2 or 3, would require an initialisation or programming time, which will be lost on the changeover, aside from the reorganization of the stations. Changes that the FMS undergoes, would require that a traditional quality management system be reprogrammed and reintegrated into the FMS, thereafter. This, of course, would mean that production is interrupted, even postponed, until the traditional quality management system is reworked.

1.3. RESEARCH AIM AND OBJECTIVES

The aim is to design, construct and evaluate an Intelligent Quality Management System, which may adapt to the change of the flexible manufacturing system. The QMS will interface with an FMS, with the purpose of collecting the various data from the FMS, in whichever order it comes in and, furthermore, make sense of it for the person at the end of the line, who receives the product after assembly. The objectives, to achieve this aim, are:

- To review the literature on Flexible Manufacturing Systems, focusing on design and operation, as well as drawbacks that arise where quality management is concerned.
- To design an Intelligent Quality Management System for the FMS, catered around the interface/communication with the FMS, the easy initialisation of the iQMS system and transition from learning, as well as application of new parameters. The aim is to design a system that matches the flexibility of the FMS and/or, handles the data requirements, without compromising the productivity.
- To develop an iQMS that caters to the unique data handling requirements of the FMS, to successfully gather, analyse and log data generated during the assembly process. The aim is to provide an innovative system that approaches the collection and use of training data differently.
- To implement a test bench for the iQMS, presenting the data collection and analysis results of the system. The aim is to verify the correctness or dependability of the developed system.

1.4. RESEARCH METHODOLOGY

The objectives of the study are realised through the following methodology:

- Literature review: A detailed review of literature on the FMS is carried out, to define flexibility and identify the advantages, as well as disadvantages associated with the implementation of quality management for FMSs. This review looked at product flexibility, as well as adaptation to production demand, related to industrial manufacturing, that aims to benefit from the re-configurability of FMS.
- System Design: The design process looked at satisfying the needs of an assumed FMS, linked to a case study based on the assembly of Circuit Breakers. Cost considerations, as well as communication, I/O requirements and software management, should be identified through the analysis of the case study, resulting in the identification of the hardware and software components that are used to construct the system.

- System Development: Hardware and software components are assembled into a iQMS, to collect FMS data. The intelligence of the system is introduced by the hardware nodes, ARDUINO microcontrollers and the C# programmed control system. The control system is in charge of the communication, data flow and analysis of the iQMS.
- System Testing: FMS data generation and simulation is used to verify the iQMS function of data collection and analysis in the learning and operational modes of the system. Learning mode shows the use of QR codes to gather training data for the introduction of a new product to the system which is close to being assembled by the FMS. QR code data decoding and processing, qualifies the ease of acquiring new product data parameters for a quick initialisation. The sample QR codes for the resulting products, will be generated and scanned to the iQMS, which will log and analyse this to categorise and define the table structure. Thereafter, the system will be put in run mode, to collect simple circuit breaker assembly data. A similar process of letting the system gather data for the various products that may be assembled and, afterwards, retrieving and looking at the logged data, to see how the iQMS handled the process. Three things may prove the concept or indicate the success of the concept, namely, the correct creation of a table based on the data retrieved from the QR codes, correct logging of the product data to the table during run mode and the final product correctly qualified as either PASS or FAIL.

Figure 1.2. below, illustrates the iterations of the research model design process. It is not enough to identify the problem, a clear definition is required, followed by the motivation, which is linked to a study of the FMS. Speculation follows from the initial process, which prompts the imagining of a solution. Theory informs the design and development process and transforms the imagined solution into a functional system or, the initial version of the system. Equipped with the application knowledge of the design model, leads to the demonstration process. Demonstrate the use of the system to solve the problem. The evaluation process informs the iteration of the system. The efficiency and effect of the application to the problem is under scrutiny. How well does the solution support the problem? The answer to this question decides whether the solution is sufficient or needs improvement.

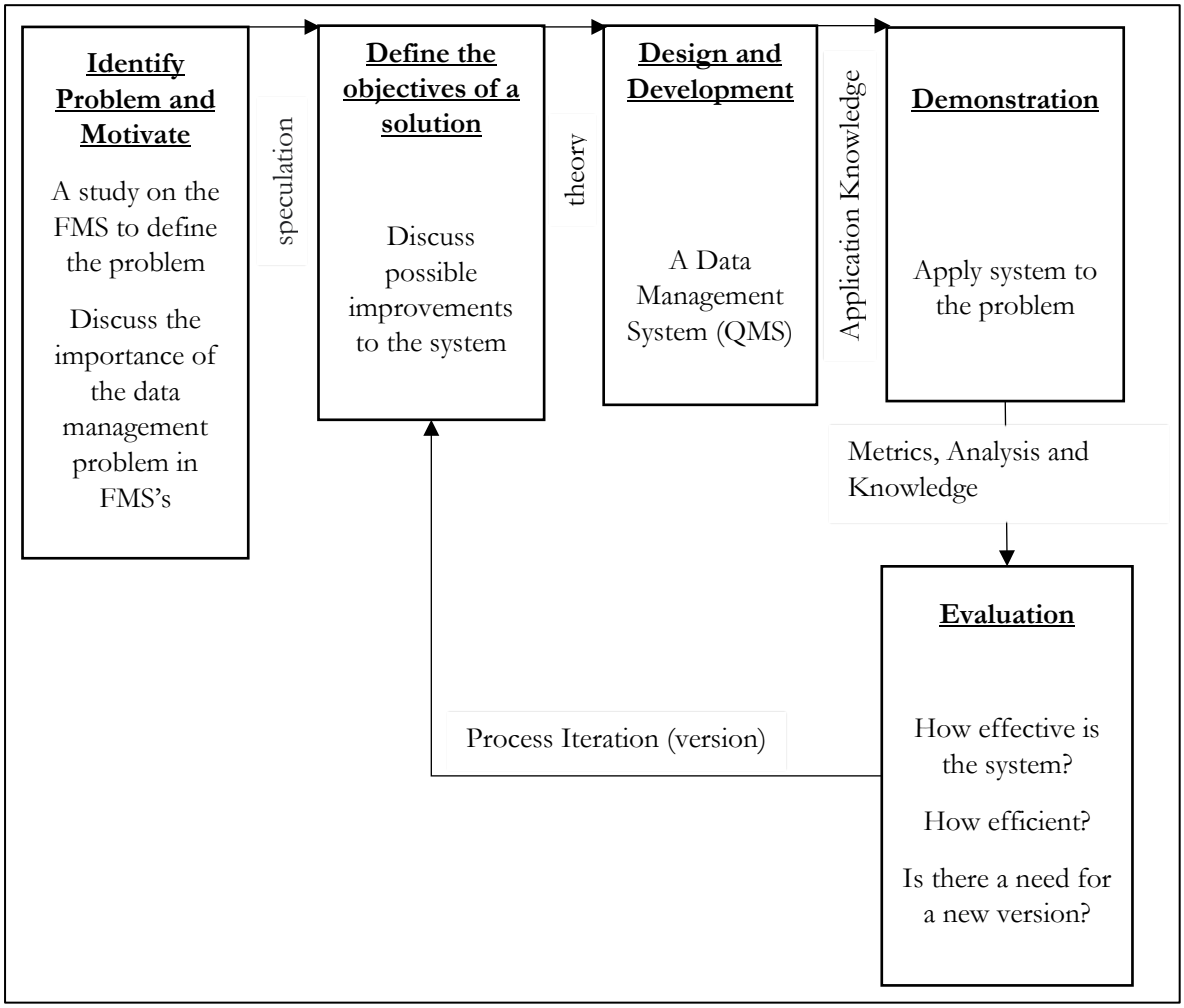


Figure 1.2: Research Design Methodology of QMS

1.5. CONTRIBUTION TO KNOWLEDGE

The main contributions of the study, in comparison with further available research regarding in the manufacturing industry and industry 4.0, are as follows:

- An innovative and intelligent data management process, that caters to the flexibility and unique data management requirements of Flexible Manufacturing Systems.
- A quality management research study in the era of Industry 4.0, highlighting how quality management practices may adapt to new digital advances in the manufacturing industry.

1.6. HYPOTHESIS

The design and development of an innovative QMS, that is able to handle and adapt to data requirements of a FMS, as it reconfigures to meet changing product and production demands.

1.7. DELIMITATION

The study was conducted with the following limitations:

- Available FMS product data was limited to a simulation that was based on the case study presented in the respective Chapter, since the study was focused on the collection and processing of data, QMS and not on the design and development of a FMS.

1.8. THESIS LAYOUT

This Thesis has been divided into five chapters.

Chapter 1 provides a background on the research, the problem statement, aim and objectives, the hypothesis, as well as the methodology.

Chapter 2 is a review of the available literature on the fourth industrial revolution, flexible manufacturing systems and quality management, as applied in manufacturing industries. The literature on flexible manufacturing is used to provide context and highlight the problem statement.

Chapter 3 presents a methodology that should highlight the assumptions that led to the formulation of the project, as well as provide a concise description of the study area and the methods employed.

Chapter 4 describes the development of the Quality Management System, detailing the hardware and software aspects of the research study. The details on the use of the ARDUINO platform, to construct the QMS's Hardware Nodes, are included in this Chapter. Furthermore, QMS Software development and deployment details are also incorporated.

Chapter 5 provides a conclusion of the research study with a discussion regarding the results, contributions of the study, as well as setting the course for prospective future studies.

CHAPTER 2: LITERATURE REVIEW ON FLEXIBLE MANUFACTURING SYSTEMS AND QUALITY MANAGEMENT

2.1. INTRODUCTION

The manufacturing industry is the basis of a nation's economy and powerfully influences the populations livelihood. Emerging technologies may have a major impact on manufacturing models, approaches, concepts and businesses [13]. The manufacturing industry forms the basis of many national economies, which is reflected in its high share of total output, employment and revenues, as well as in the creation of sustainable economic growth [14]. In recent decades, producers and suppliers of goods and services have improved the quality of their organizations, using innovative technologies. This is in view of the fact that the industry is undergoing transformation and evolution towards complete digitization and the intelligence of production processes, to ensure high efficiency [15].

The manufacturing industry underwent four main revolutions. The First Revolution spans from the end of the 18th century to the beginning of the 19th century and involved the emergence of mechanization**.** Nearly a century later, the Second Revolution was sparked, when new technological advancements initiated the emergence of new sources of energy: electricity, gas and oil. As a result, the development of the combustion engine set out to use these new resources to their full potential. A Third Industrial Revolution appeared with the emergence of a new form of energy, with the potential of surpassing its predecessors: nuclear energy. This revolution witnessed the rise of electronics, with the transistor and microprocessor, as well as the rise of telecommunications and computers [16]. Presently, we are on the brink of a new industrial revolution, namely, the Fourth Industrial Resolution or Industry 4.0. We have reached a point where a clear application-pull (industrial demands) exists and technology-push (technological advances), acting together as a driving force for this new revolution [17].

The literature review encompasses research on the revolution of the manufacturing industry, the specific shortfalls of Flexible Manufacturing Systems, related to the study and highlights the quality management problem, addressed in the study. The methodology of the research is largely influenced by the scope of the study, which is focused on improvements to quality management in an evolving manufacturing industry. Flexible Manufacturing Systems are the latest iteration in the production methods employed in the manufacturing industry. The study makes use of an experimental and simulation-based research methodology.

2.2. INDUSTRY 4.0

Industry 4.0, is a name for the current trend of automation and data exchange in manufacturing technologies. It includes cyber-physical systems, the Internet of Things, cloud computing and cognitive computing. Industry 4.0 is commonly referred to as the Fourth Industrial Revolution. Subsections of Industry 4.0 are flexible manufacturing systems and quality management [18].

2.2.1. Flexible Manufacturing System(s)

A flexible manufacturing system may be defined as a computer-controlled production system, capable of processing a variety of part types. Flexibility characterizes production ability to be readjusted to the multipart, small or medium-batch production. Various flexible manufacturing systems exist. FMS are categorised according to the targeted process being improved. These categories are as follows [19] [20]:

1. **Variant flexibility**: Ability to manufacture or assemble more variants of a product. (Product flexibility)
2. **Quantity flexibility**: Ability to adapt the production system to fluctuating volumes.
3. **Technology flexibility**: the ability of manufacturing and assembly systems to be used for a number of technologies.

4. **Successor flexibility**: ability to, furthermore, use equipment or parts for future products.
5. **External flexibility**: ability to change the system by exchanging elements (example: replacing robot gripper).
6. **Internal flexibility**: ability to change the system without modifications (example: automatic tool changes).
7. **Personnel deployment flexibility**: ability to operate with more or fewer employees, with various qualifications.

This intelligent manufacturing, takes advantage of advanced information and manufacturing technologies to achieve flexible, smart and reconfigurable manufacturing processes, addressing a dynamic and global market. It is regarded as a new manufacturing model, based on intelligent science and technology, that greatly upgrades the design, production, management and integration of the whole life cycle of a typical product [21]. Production efficiency, product quality and service level may be improved [22].

Flexible manufacturing consists of concepts, such as Intelligent manufacturing, Cloud manufacturing and IoT-enabled manufacturing [23]. There are similarities when looking at the aims of intelligent/smart decision-making in manufacturing systems and the optimization of various manufacturing resources [24]. Table 2.1. illustrates this, with a comparison of three manufacturing concepts.

Table 2.1: Comparison of the Three Manufacturing Concepts

Concepts	**Major Characteristics**	**Supporting Technologies**	**Major Research**	**Applications**
Intelligent Manufact-uring	• AI-based smart decision making • Advanced automotive production • Adaptive and flexible manufacturing systems [24]	• Big data processing • Advanced robotics • Industrial connectivity services •Last-generation sensors [25]	• Advanced manufacturing decision-making models • Human-machine integration • AI-enabled machine learning • Machine-to-machine connectivity [26]	•A smart manufacturing system with a portrait of an ISO STEP tolerance standard •A product life-cycle test bed enabling intelligent manufacturing •Agent-based IMSs [27] •Intelligent manufacturing planning and control systems [28]

IoT-enabled Manufact-uring	• Auto-ID technology-based smart manufacturing system • Real-time data collection • Real-time visibility and traceability of production processes • Real-time manufacturing decision-making [29]	• IoT • Wireless production • BDA • Cloud computing [30]	• Real-time data-driven decision-making models • Real-time data visualization • SMO modelling • Models of SMO behaviours [31]	• An RFID-based resources management system • An IoT-enabled smart construction production system [32] • An RFID-based job shop WIP inventories management system [29] • An RFID-enabled real-time production planning and scheduling system [31]

Cloud Manufact-uring	• Manufacturing service distribution and sharing • Intelligent capability management • Manufacturing cloud service Management	• Cloud computing • IoT • Virtualization method • Service-oriented technology [33]	•Modelling of manufacturing resources and capabilities •Manufacturing services configuration •Manufacturing cloud architecture	• Data visualization in a cloud manufacturing shop floor • QoS-based service composition selection in a cloud manufacturing system [34] • Smart cloud manufacturing using the IoT • A semantic web-based framework in cloud manufacturing

In the Industry 4.0 era, an Intelligent Manufacturing System (IMS) uses Service-Oriented Architecture (SOA), via the Internet, providing collaborative, customizable, flexible and reconfigurable services to end-users, thus enabling a highly integrated human-machine manufacturing system [35]. This high integration of human-machine cooperation, aims to establish an ecosystem of the various manufacturing elements involved in IMS, ensuring that organizational, managerial, and technical levels may be seamlessly combined. An example of IMS is the Festo Didactic cyber-physical factory, which offers technical training and qualifications to large vendors, universities and schools, as part of the German Government's Platform Industry 4.0 strategic initiative [36].

The flexibility of these systems is limited only by their parameter of geometry, power and function, however, within these constraints, they may perform a virtually infinite variety of tasks [37] [38]. Flexibility allows the adjustment of the manufacturing system within a defined flexibility corridor [39]. In this sense, flexibility describes the ability of a production system to adjust the manufacturing system rapidly, with a low cost. Changes are defined through predefined packages of measures and are limited by certain flexibility corridors at the time of planning [40].

2.2.2. Quality Management

A Quality Management System is a set of policies, processes and procedures required for an organisation to meet its objectives and continually improve its capabilities [41]. Product and environmental quality are critical to the welfare of human beings. Recent research on QM in the international Journal of Production Economics, has been connected to many further domains, such as impact on innovation, company performance and corporate social responsibility [42].

Mixture data management, together with tool management, forms an important unit in FMS. Both tools and fixtures have a significant effect on product quality. This is particularly important in FMSs, in view of the fact of the high-quality requirements. Unmanned use of said systems, necessitates control, based on good data management [43].

Regarded as the main contributors to various environmental problems, firms are facing tremendous pressure from the government, consumers, media, environmental non-government organisations and further stakeholders, to incorporate both quality management and environmental management within their business practice [44].

The need for more flexible and efficient data management, in manufacturing systems has become obvious. To secure the highest utilization rate and maximum productivity of manufacturing systems, it is necessary to be able to acquire the correct information, at the right time and place. Non-production and inefficiency in single operated, stand-alone NC-machine tools and other NC-equipment, are due largely to human fault: elapsed times

between the receipt of information, the formulation of a decision and the performance of the command [45].

A typical automation system would be tasked with assembling and verifying the assembly of that product. This would involve certain data being processed by the system, such as the digital and analogue data retrieved from proximity and other sensors. Image data from the quality inspection performed by the machine vision station(s), as well as prompts and messages sent back and forth between the various stations, the assembly management Information Technology System and the database system responsible for archiving any of the data designated to be logged for future use [46].

Operational control of an FMS is exceptionally complicated and involves accessing large static and dynamic data sets; representing machine configuration and characteristics, system status and process plans and complex control algorithms. The control algorithms are structured hierarchically, where an upper-level issues commands to a lower level and obtains feedback on the achievement of these commands.

2.3. INTERFACING THE FMS AND QMS

To construct an innovative, Intelligent Quality Management System for Flexible Manufacturing Systems, sensors should be connected from the Flexible Manufacturing Systems to the Quality Management System.

There are several different means of integrating the information provided by multiple sensors into the operation of a system. The most straightforward approach is to allow the information from each sensor to serve as a separate input to the system controller [47]. This mentioned approach of treating each sensor as an individual input to the system controller, is precisely the form of logic that should be employed, to manage the communication and data exchange between the FMS and the QMS under development.

To connect sensors to the Quality Management System, one should consider hardware interfaces. Arduino is an open-source electronics platform, based on easy-to-use

hardware and software. A microcontroller consists of a microchip on a circuit board, with read-write capabilities, memory, inputs and outputs [48].

Arduino boards may read inputs; light on a sensor; a finger on a button and turn it into an output; activating a motor; turning on an LED; publishing something online. One may communicate command to your board, by sending a set of instructions to the microcontroller on the board. To do so, you use the Arduino programming language, based on wiring and the Arduino Software (IDE), based on processing.

Arduino has been used in thousands of various projects and applications. The Arduino software is user-friendly for beginners, yet efficiently flexible for advanced users. Arduino further simplifies the process of working with microcontrollers, however, it offers a few advantages over other systems. The two advantages that stand out for the purpose of the proposed production information system, are the cross-platform application of the Arduino, as well as the simple and clear programming environment [49].

2.4. RELATED RESEARCH FINDINGS

Nylund et al [50], developed a learning environment for engineering education that focuses on planning, operation and analysis of Flexible Manufacturing Systems (FMS).

Somasundaram et al [51], takes advantage of the technology, such as Artificial Intelligence/Machine Learning to use past historic data in an institution and current data of student profiles/performance, which may be used to analyse and predict learning gaps and suggest the learning steps a student should take, to improve his/her performance. The Measurement and Analysis is carried out on all the processes for process compliance and process effectiveness. Customer satisfaction is measured, and continuous improvement is planned and achieved.

Gopi et al [52], implemented a study to solve distributed Scheduling problems, using discrete event simulation approach. The objective is to determine the allocation of tasks to factories and schedule the production process in each factory, under two cases, with and

without breakdowns in the machines. In this model, alternative production routings are not considered.

Sanchez-Marquez et al [53], developed a methodology that allows practitioners to identify and quantify the cause-and-effect relationships between internal and external metrics, to move from performance measurement to effective performance management, by using key performance indicators (KPIs) of the balanced scorecard BSC.

Mahmood et al [54], analyses the performance of a FMS through manufacturing process modelling and simulation. Integration of system dynamic analysis, with reliability estimation methods, enable engineers to note the most unreliable places in a production process and supports the decision making for reliability improvement of a production system.

Testa et al [55], discusses the Quality by Design (QbD) based strategy and operation of an end-to-end integrated continuous manufacturing (ICM) pilot plant, that produces both a small-molecule active pharmaceutical ingredient (API) and oral solid dosages (OSDs).

Lugaresi et al [56], uses simulation models, to replicate the behaviour of a real production line, by moving parts such as spheres or discs along an appropriate route and reflect operation times, by letting parts wait in a station, for an appropriate time span. The results testify that the proposed lab-scale models may be used successfully, to test production planning and control approaches.

Bi et al [57], investigates the impact of Enterprise Architecture (EA) on system capabilities in dealing with changes and uncertainties in globalised business environments. The Shannon entropy is adopted, to measure the complexity of systems and illustrate the roles of EAs in managing system complexity and achieving system stability, in the long term.

Singh [58], examines the integration of Big Data (BD), Industry 4.0 and Cyber-Physical Systems (CPS). The research addresses the state-of-the-art technologies and phases, to bring attention to data integrity, quality, privacy, availability, scalability, transformation, legitimacy and monitoring issues, as well as governance.

Luo et al [59], contribute, by proposing an architecture realising automatic modelling and data-driven simulation, first in the cloud simulation environment and filling the gap of dynamic resource allocation in the research of Automated Flexible Production Lines (AFPLs).

Zonnenshain et al [60], found that in the last few years the quality discipline went into stagnation. Proposals for innovative models have deteriorated.

Carvalho et al [61], identified the growing need for quality management practices as more Industry 4.0 technologies are introduced to manufacturing.

2.5. FINDINGS AND SUGGESTIONS

After reviewing the main available publications on FMS and Quality management, the following was observed:

- Available studies focus on the virtual implementation of the FMS, in order to investigate and improve on system optimization and analysis of manufacturing industry financial benefits.
- The majority of studies have used the study of quality management systems, looking to improve customer satisfaction.
- The available studies point towards the use of FMS and quality management methods, to facilitate the teaching and learning process.
- The majority of available studies, revolve around the management of tools and parts, as well as improving system scheduling and performance.
- The available studies are mainly looking at reducing the initial costs that are synonymous with the conversion of a standard manufacturing systems to a FMS.
- Available research is looking into the application of principles learned from FMS's to different industries, to improve product development, for commercialization and investigating future challenges.

- Available research is largely concerned with using the developments in the manufacturing industry and related research, to enhance decision-making and reaction time.
- The research shows a lack of quality management practices, to cater to the increase in Industry 4.0 advances in the manufacturing industry.

Suggestions responding to the above findings, are proposed as follows:

- Develop a quality management system, that caters to the information management requirements of a Flexible Manufacturing System (FMS). The requirement for data collection and analysis, where FMS's are concerned, is a unique challenge because of the unorthodox approach to product assembly and quality assurance.
- Innovate the way training data is collected, e.g., use barcodes as a medium to train the QMS, by creating learning data tables. Deploy the learning data tables in the task of qualifying product data.
- Retain a level of user input/control in the operation of the QMS.

CHAPTER 3: DESIGN OF THE QUALITY MANAGEMENT SYSTEM (QMS)

3.1. INTRODUCTION

Chapter 3 is divided into three sections and, then, rounded up by a summary. The first look is towards related research, that forms the base of the methodology for the study. That is followed by looking at the case study and the research strategy. A discussion on the research design follows, aimed at forming a comprehensive idea, based on the case study and methodology discussed and, furthermore, elaborates on the data collection and analysis tool designs.

3.2. RELATED RESEARCH

Literature showed that there is a requirement for more flexible and efficient data management, in manufacturing systems. The right information at the right time and at the right place, is a necessity, to secure the highest utilization rate and maximum productivity of manufacturing systems [62]. Non productivity and inefficiency in single operated, stand-alone NC-machine tools and other NC-equipment, are due largely to elapsed times, caused by human processing from receipt of information to formulation of a decision and the performance of the command. The advantages of FMSs are obvious, in the case where they are compared to stand alone machines. FMS are designed mostly on the idea of centralized control. These types of systems are challenging and expensive to expand and maintain. Centralization results in the FMS being dependent on a network and a main computer, that may cause inconveniences with breakdowns [40].

The fundamental building block of an FMS is data communication, as flexibility is mainly imparted by integrating the functions of various elements, such as machining cells, robots and AGVs using computers. FMS are particularly complex and expensive systems and require a large amount of planning and investment. The majority of the published

literature addresses the operational design issues (such as loading, sequencing, and/or scheduling strategies under which to operate the facility), whilst the equipment design has gained relatively limited attention. Various approaches and techniques are employed in the design of FMSs, to deal with the challenges that are commonly associated with them. Analytical approaches, based largely on mathematical relationships, often result in inefficient algorithms. This inefficiency is linked to the size of the FMS; the larger the system, the more time computation takes and the higher the cost [63].

In cases of complex production setups, such as FMS, it is important to develop tools to support the decision-making, where changes to the production processes not only cope with the new requirements but maximise value and maintain profitability of the systems. Simulation is one of the tools proven to support the decision-making for flexible systems, in real life applications [64] [65]. The key element of successful decision making is through simulation tests in accurate data collection and interpretation [66].

This research focuses on demonstrating a methodology for innovating data management in a FMS. It further looks at ensuring that the quality of any product, assembled or developed in a FMS, is maintained at a standard that is acceptable, based on the set parameters. The previously examined material, highlights two important points, namely, data communication and simulation.

3.3. METHODOLOGY

This study is constructed around the complexity and flexibility of FMSs, with the focus being on data handling and data communication. Product assembly, executed by the FMS, generates data at stations specific to quality assessment. Structural, Mechanical and Electrical integrity requires verification. Structural Integrity relies on the image processing stations, whereas Mechanical and Electrical stations are set up differently, depending on the product in development. Mechanical integrity is assured by a station's setup, to test strength/stress and dynamic/static mobility, as well as longevity of parts/joints/contacts. Electrical integrity stations are set up to test for current, voltage, resistance and/or conductance. Temperature often forms part of the testing parameters.

A typical FMS will be centred around the product, as well as the effort to gain as much advantage in the production process as possible. This advantage may be gained in the speed of the assembly/development process or in the scaling of the system itself to offer variations of the product. The manufacturing flexibility may further be explored, where the aim is for the system to adjust from one product that might be outdated to a newer product. Products undergoing an assembly process are linked to data that is specific to the final operation of the individual product. The product data gathered in the process, represents a product life cycle record. The specifics are dependent on the different parts that the product consists of the tests conducted through the assembly process and the progress as compared to set production targets.

The QMSs main goal, is to process all the data, however, this is dependent on the mode of operation that is in execution at the time.

The following information at each stage, is crucial to the functioning of the QMS:

- **Mode** of operation - Which mode of operation is the system executing? The QMS has two modes of operation: RUN and LEARN.
- **QR Code** Scanning - Which type of QR code is being scanned? Type being either PASS or FAIL.
- **Data** Handling - How should the data be processed for the current operation? Again, there are two possibilities: CREATION or CORRELATION.

Simulation, as a tool, is used in the study for testing and benchmarking the QMS. Essential to this task is Input/ Output simulation, both digital and analogue data, moving between the FMS inspection stations and the QMS.

The development process considered many aspects of a data management system. The process of data management includes a combination of various functions that collectively aim to ensure that the data in the system is accurate, available and accessible. The functions are data collection, validation/processing and storage. The following is a brief discussion on how each of these functions are incorporated into the project.

- Data Collection:
    - In LEARN mode, Quick Response (QR) codes are used, to obtain data associated with the product. A minimum of three QR codes are required for each condition of the product, i.e. FAIL or PASS condition.

- In RUN mode, data is requested from the hardware nodes: ARDUINO microcontroller circuit. The connected Digital and Analog I/O responds to the QMS with corresponding data.

- Data Validation/Processing
  - The QR code should be broken down, to identify the values/variables; these values are product specific. The order of the values and the conditions associated with those values, correlate to the assembly process viz. the assembly stations. Quality control hinges on the values to determine whether a product PASSED or FAILED the process of assembly.
  - Hardware Nodes communicate the Digital I/O states and Analog I/O values to the Control System. This is in response to a request made by the Control System.
- Data Storage
  - Processed QR code data is gathered during the LEARN phase, to instruct the system and prepare for the correlation and verification process. Correlation and Verification occurs during the RUN phase, to determine the PASS or FAIL conditions.
  - Processed product data is stored in the database. Each product is assigned an overall state of PASS or FAIL, accordingly.

A common thread to all three aspects considered during development, is communication. Communication should continuously be maintained during the assembly and quality assessment process, to ensure the success of the QMS' data capturing.

3.4. CASE STUDY BASED STRATEGY

The case study for the project, highlights the data processing aspects as they will be tackled by the QMS, with data flowing from different stations of a manufacturing system. Stations, in the general sense of the study, refer to the assembly and inspection stations. However, the manufacturing system by itself would serve no clear goal without the product

that is to be observed in the assembly or development process. The product, in this case, being the Circuit Breaker (CB).

The CB presents as a product with possible variations and the manufacturing process involved in the realisation of the possible complexities forms the basis of the case study. The CB is a product consisting of different units/parts, that contribute to the final product function. The placement of these parts warrants a vision inspection, to ascertain the accuracy of the assembly. This means that the function of the final product requires verification, however, the product assembly process cannot be ignored to be verified solely at the final stage of development. Constant validation of assembly accuracy needs to take place so that, if possible, adjustments may be made. Validation of the data collected in the process of assembly, inspection and testing, proof of both product quality and the manufacturing system's quality standards. Quality standards are linked to the consistency of the manufacturing process being carried out.

Product assembly, in this case, will be based on the parts that produce the final product and the function of the product. Part placement and part movement are linked to the part being introduced to the product at the different points of assembly. Product function is a measure of the product quality, as well as the manufacturing process correctness. Different processes of experimentation and simulation may reduce the actions required in a manufacturing system, down to simple tasks/scenarios.

In the case study, there are three different scenarios to be expected from the FMS side:

- QR code Scanning
- Image Inspection
- Electrical Testing

It should be made clear that this in no way means that solely three inspection stations may be in the entire system. The FMS may have several stations; if required, that are tasked with the above, not to mention the stations that are mainly tasked with assembling the product.

The optimal way to present the case study, would be with a simple diagram, comprising of the different larger units first and then, proceed to break it down into the smaller and simpler units. The FMS has received attention in previous discussions and the

figure below just highlights the positional relationships between the FMS and QMS, with an indication that data may be bidirectional between the systems.

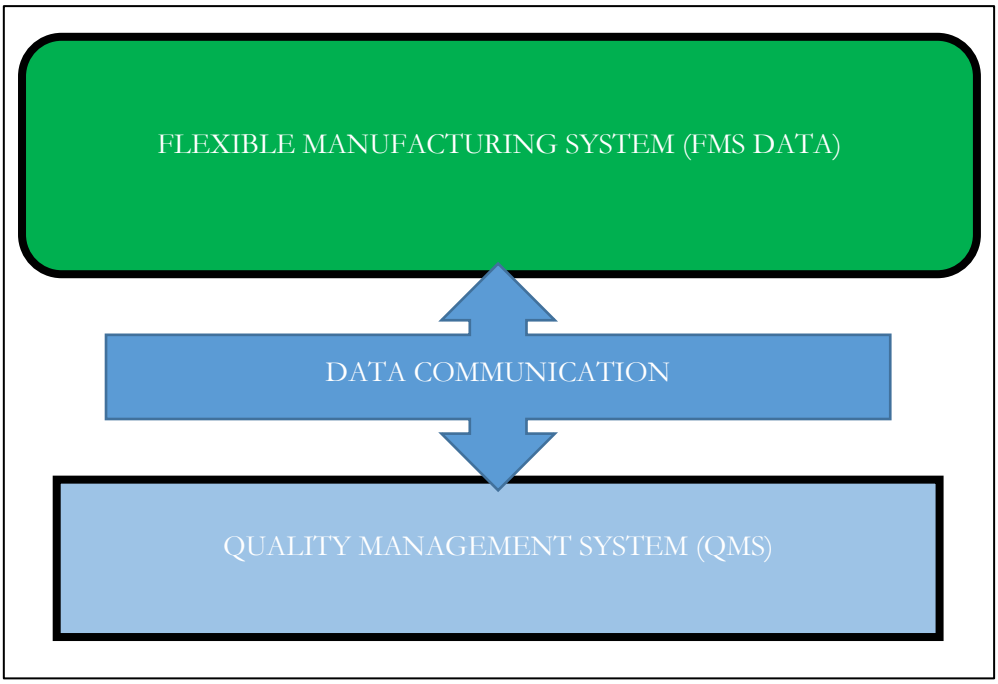


Figure 3.1: Prospective Data Flow from the FMS to the QMS

Since the bulk of the work for this research is on the QMS, this Section focuses on the arrangement or, rather, operational organization of the system. The QMS is made up of the Control Software, Archive (Database System) and the Communication (I/O Peripheral) unit. System operators interact with the Control Software, with the remainder of the system setup to follow pre-set procedures, for a successful system execution.

Previously there was a flowchart to indicate the envisioned operation of the QMS, in conjunction with the FMS. Figure 3.2, below, concentrates on the QMS and the interaction of the various parts to the Control Software.

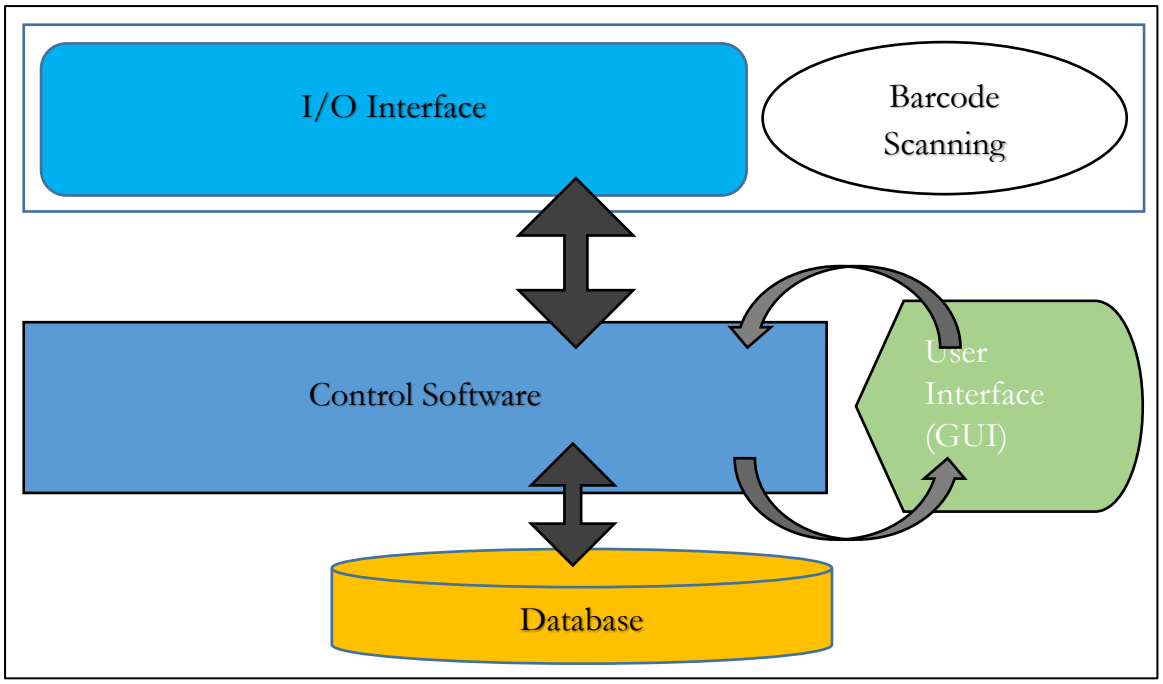


Figure 3.2: Interaction of the Parts of the QMS

The interactions between the various units of the QMS, are bidirectional, flowing back and forth. The QMS relies on correct and constant data communication, to fulfil the purpose that it is developed for.

3.4.1. Mode(s) of Operation

There are two modes of operation, RUN and LEARN mode. The System Operator is tasked with selecting the mode of operation, depending on whether the FMS is close to manufacturing a new product or, whether it is continuing from where it stopped. RUN mode is selected to continue the manufacturing process that is presently underway. LEARN mode is selected, introducing a new product manufacturing process to the FMS.

3.4.2. QR Code Scanning

Product assembly is often assisted using barcodes, which keep record of the specific product being manufactured. These barcodes may be generated at the end of a successful product assembly process, to label the product. Depending on the type of product, the

information varies from the date of manufacture to all vital specifics that may be required by quality assurance companies. Barcodes can be one dimensional (1D) or two dimensional (2D). The defining difference being that data is encoded either linearly for 1D barcodes and vertically, as well as horizontally for 2D barcodes. This encoding allows 2D barcodes, such as the QR code, to hold more data and still occupy less space if required.

QR Code scanning in this study, depends on the mode of operation selected. Scanning a QR Code after the system operator has initiated, the QMS in RUN mode, may result in a standard quality assurance routine; that is, receiving data from the FMS and archiving it for that product, with a unique ID. Additional result from the inspection stations, may be appended to the product data and, at the end of the process, a final QR code may be generated. Product information correlation and analysis is performed on the archived data.

QR code scanning during the QMSs LEARN mode, is carried out when the system requires reprogramming for a new product. Data may be extrapolated from the QR code and used to create training data tables, which set the parameters for the new product that will be manufactured/assembled. Slight variations in the data linked to a product, does not disqualify a product, unless the data falls outside set parameters. Parameters may be set to define a maximum and minimum value for the different product data variables collected, as the product undergoes the manufacturing process.

3.4.3. Data Handling

Product specifications are important for quality assurance purposes. Product reports and/or specifications sheets are compiled, to provide enough information on the product. Data generated and collected during the manufacturing process is used to compile reports on product specifications.

QR codes may be used to collect training data and to generate training data tables. The training data tables will be used in the product quality assessment. Product assessment will qualify the production process for the product undergoing the assemble process. A data correlation may be performed on the product data and the data collected during the learning process. Correlation involves the use of stored data of *previous* products, in conjunction with the *current* product data, to decide on the status of the product. The

product status decision may be FAIL or PASS. The training data table creation relies on the data extrapolated from the QR code. The results from this data handling process, qualify the product as a success or failure. The data tables are created using SQL server queries generated, based on the number of fields that may be identified from the data. The table will be specific to this product and may be based on several QR codes scanned, for the purpose of teaching the QMS (LEARN mode).

3.5. DESIGN AND OPERATION

Data dependent systems are designed on the back of the database and data flowing from the FMS. A Just-In-Time (JIT) approach is used for the design and, furthermore, to observe the operation of the system. JIT simply means that the system should be centred around handling the correct data, at the right time and make timeous decisions, based on this data.

The following quote served as a guide in the development of the methods and design:

- Operational control of an FMS is particularly intricate and involves accessing large static and dynamic data sets and complex control algorithms. The control algorithms are structured hierarchically, where upper-level issues commands to a lower level and obtains feedback on the achievement of these commands [40].

Table 3.1: Project as Seen from an Event and Status Point of View

CONTROL SOFTWARE EVENT	***STATUS***			
Mode of Operation	RUN		LEARN	
QR code Scan	N/A Assembly continues	N/A Assembly continues	PASS	FAIL
Data Handling	CORRELATE		CREATE	
Decision/ Feedback	PASS	FAIL	SUCCESS	FAILURE

Table 3.1 places the project into perspective, by simplifying it into the EVENT and STATUS columns. EVENT, in regard to the project, focuses on the Control Software and the various data driven events that are expected, as explained in the previous Section. STATUS, in relation to the EVENT, guides the Control Software in issuing commands and prepares for feedback, which initiates decision protocols.

The Control Software will initiate the different EVENTs and then, the following may determine the STATUS of the EVENT. The Mode of Operation EVENT is the one in which the System Operator may have direct input, to initiate either the RUN or LEARN modes. The EVENTs, thereafter, have set parameters and protocols, to initiate decision making and actions relating to the decisions.

Data simulation will be incorporated into the system, to demonstrate the interface of the QMS to an FMS; simulation of digital and analogue I/O data and QR code scan and print.

3.6. MATERIALS AND METHODS

The QMS is a project which makes use of the following software platforms: Visual Studio 2015, ARDUINO 1.8.16 and MySQL. All of these are powerful tools on their own and lend their "strength" to the project, as it may be apparent in the following brief description.

Visual Studio is a comprehensive Integrated Development Environment (IDE), for .NET and C++ developers. The main control of the project is developed on this platform.

ARDUINO is an open-source electronic prototyping platform, enabling users to create interactive electronic objects/projects. A Digital and Analog I/O monitoring object is based on the Arduino Uno microcontroller board and receives instructions from the main system.

MySQL is an open-source relational database management system, using Structured Query Language (SQL). SQL is a domain-specific language used in programming and designed for managing data, held in a relational database management system. MySQL Workbench serves to construct and test queries and procedures beforehand and sets up an elementary database, which is queried/manipulated throughout the project lifecycle.

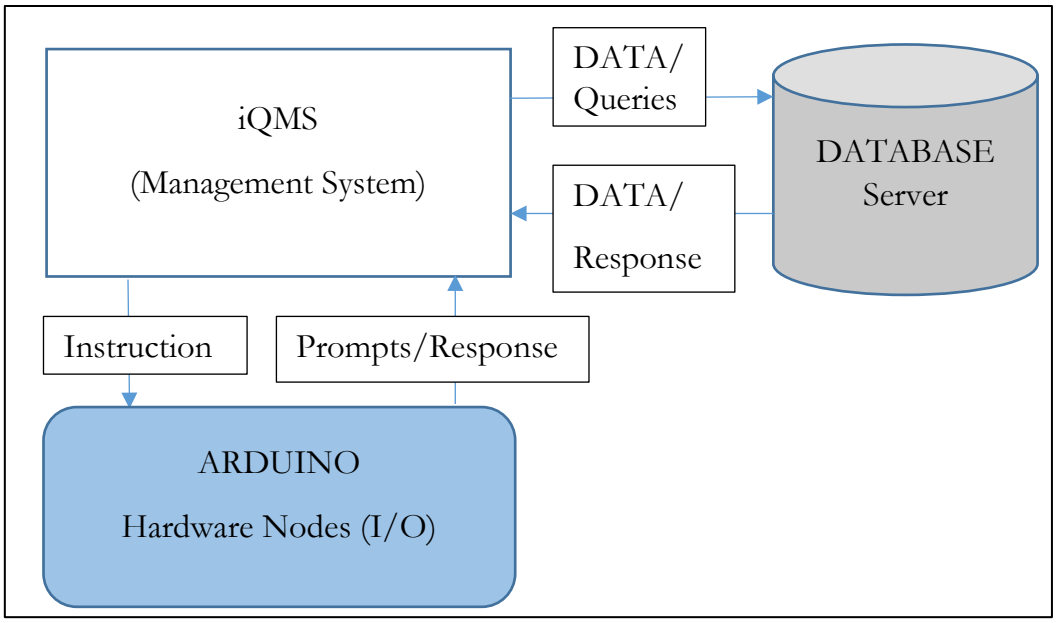


Figure 3.3: Multi-Platform System Setup

Figure 3.3 illustrates the relationship of the different platforms working as a unit, to accomplish the single task of resolving data management for a Flexible assembly system. The setup indicates two-way communication "lines" between the Management System and the ARDUINO, as well as between the Database Server and the Management System.

3.6.1. ARDUINO Hardware Nodes

The ARDUINO 1.8.16 IDE, with the aid of a built in Serial Monitor, enabled the development of the hardware node. The setup consists of the Keyestudio ARDUINO Uno microcontroller (ATmega328P-PU), with electronic components.

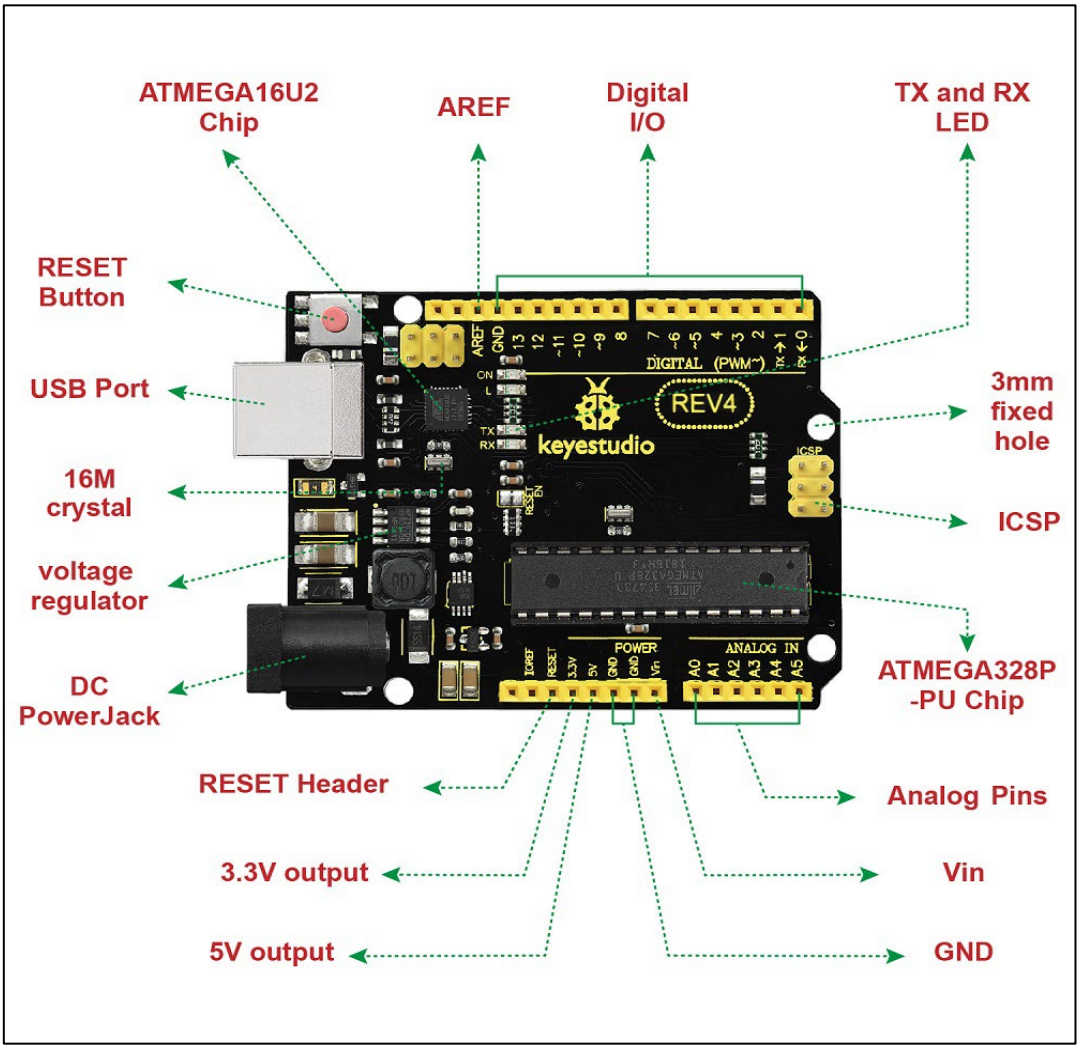


Figure 3.4: Keyestudio ARDUINO UNO Microcontroller

Figure 3.4 is an illustration of the microcontroller in use for the Hardware Node(s), with the distinctive functions indicated and, Table 3.2 below, provides the specifications of the ATmega328P-PU microcontroller.

Table 3.2: ATmega328P-PU Microcontroller Specifications

Microcontroller	ATmega328P-PU
Operating Voltage	5V
Input Voltage (recommended)	DC7-12V
Digital I/O Pins	14 (D0-D13) (of which 6 provide PWM output)
PWM Digital I/O Pins	6 (D3, D5, D6, D9, D10, D11)
Analog Input Pins	6 (A0-A5)
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P-PU)
EEPROM	1 KB (ATmega328P-PU)
Clock Speed	16 MHz
LED_BUILTIN	D13

Considerations were provided to all that the ARDUINO UNO microcontroller offers and the following proved to be the leading use of the ports. Eight Digital I/O pins are set up for each microcontroller board, 4 Digital Inputs and 4 Digital Outputs.

One Analog Input is available to the user and monitored for any changes during the assembly process. This analogue input is intended for the mechanical and/or electrical

values related to the product under development. Serial Communication was chosen as the standard to use between the ARDUINO and QMS Application.

Serial communication between the Hardware nodes and QMS is maintained through the COM ports, at a fixed baud rate of 9600. This is a standard default baud rate for serial communication, translating to a transfer of 9600 bits per second.

3.6.2. Database Server

MySQL Server offers a workbench, as mentioned previously, which enabled the basic setup of the database server. This means setting up a Local Instance, which may be used to transact with the QMS and manage data moving in and out of MySQL database server. Figure 3.5., below, presents the setup parameters used.

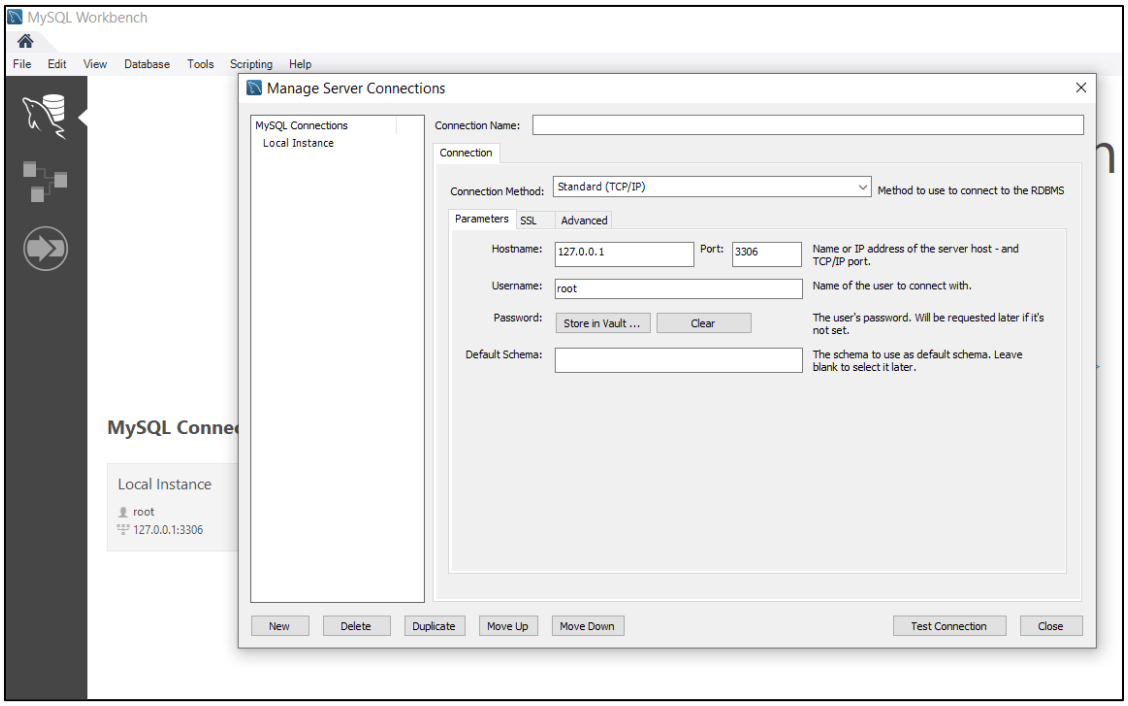


Figure 3.5: MySQL Workbench Setup Window

MySQL Workbench enables a Database Administrator, developer or data architect to visually design, model, generate and manage relational databases. It is a dashboard with visual tools for creating, executing and optimizing SQL queries.

MySQL Server attracts global attention, as it is small, fast and relatively low priced. Furthermore, a Community version is available for free download. This version is supported by an active community of open-source developers and enthusiasts.

Database linking is established through the localhost, with database schema **iqms-app**. Server Connection is set as a Standard (TCP/IP) connection, LocalHost **127.0.0.1** address and port **3306** used, with a password setup on creation and stored in the vault, avoiding constant credential prompts.

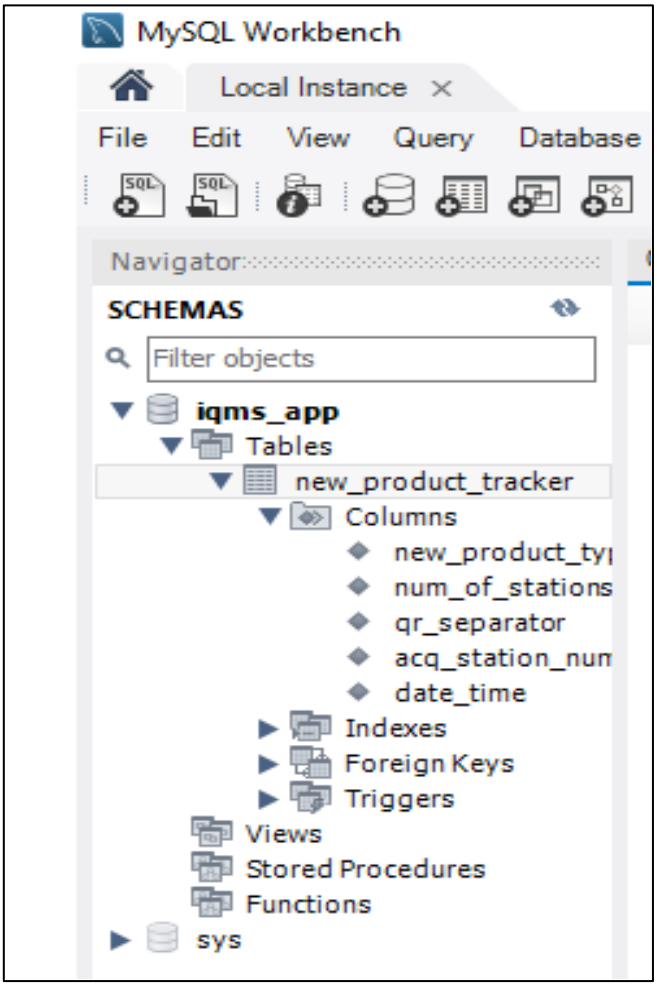


Figure 3.6: Setup of Database Schema in MySQL Workbench

Figure 3.6. illustrates the initial setup of the DB Schema **iqms_app,** to track all the new products introduced into the system. Fixed table column names are used in this case. Table columns, created when introducing new products to the system, are more generic.

MySQL Databases are inherently relational and this links to the use case. The description given of the CB as a product serves to inform the relationship that the tables will adopt, which is a one-to-many relationship. A variety of CB types will be recorded, and each type has the potential of many products linked to the same parameters, that define that type. On creating the new product tables, this factor is taken into account. The spawned tables, whether created during the LEARN or Run mode, will link to the product type tables.

3.6.3. The Quality Management System

The QMS Application was developed using the Visual Studio platform, in the C# (C sharp) programming language. System control and Data management, stem from this main application. The aforementioned Hardware Nodes and Database Server are dependent on the instructions and prompts generated by the QMS application. Communication is maintained throughout the deployment of the application.

Flexible data management is the focus of the project and, with this in mind, the application was developed to operate in two modes. LEARN mode revolves around gathering training data. The training data correlates to the configuration of the assembly system, as well as the product under assembly. Product data will define the data boundaries, that qualify the product, when assessing the quality. RUN mode is the quality assessment process, which starts out with the congregation of the present product data.

The product data that will be captured, is dependent on the station and, thus, the data type. The case study is recalled at this point, to highlight the expected data. The case study revolved around the assembly of several circuit breaker(s), assembled using a reconfigurable system. The circuit breakers are distinguished by type and have certain data

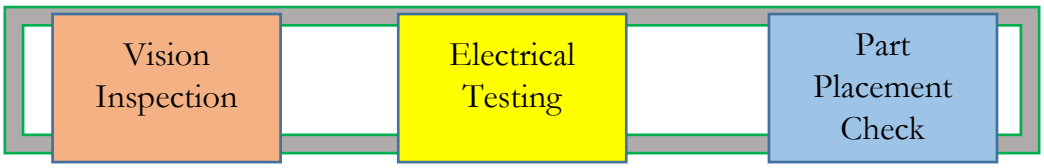
associated with the type. Data relating to this case study requires a clear understanding of this, therefore, a prompt visual discussion follows.

Circuit Breaker **A** (CB A) requires the stations to be configured for a Part placement check, followed by Electrical testing and, then, Vision inspection, shown in Figure 3.7 below:

Figure 3.7: Circuit Breaker A - Station Setup

Circuit Breaker **B** (CB B) requires the stations to be configured for a Part placement check, followed by Vision inspection and, then, Electrical testing, shown in Figure 3.8.

Circuit Breaker B - Station Setup



Circuit Breaker C - Station Setup

Circuit Breaker **C** (CB C) requires the stations to be configured for Vision Inspection, followed by Electrical testing and, then, a Part placement check, shown in Figure 3.9.

Each station produces data relating to the same product and is transmitted to the QMS through the hardware nodes. CB A data and CB B data are different in the order

received, and both are different from CB C, to a certain degree. Each circuit breaker configuration may provide a different data variable, at the specific station.

Figure 3.10 below, illustrates the two modes of operation that the QMS may operate in. The user is the operator/supervisor of the system and will follow prompts, in response to the interaction made throughout the product assembly and evaluation process.

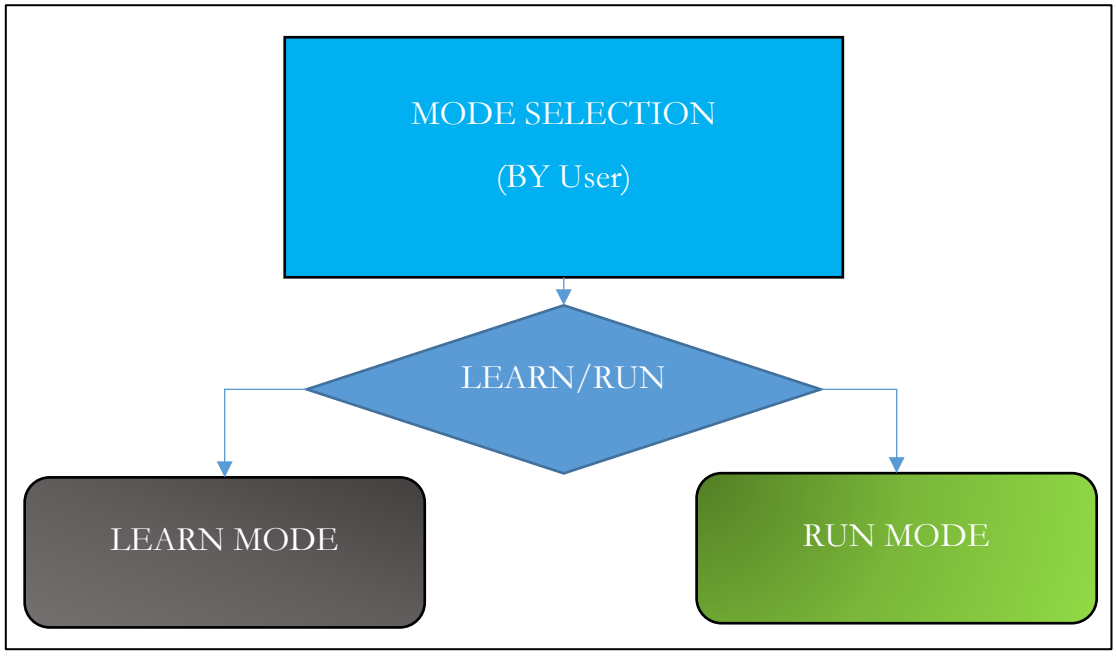


Figure 3.8:QMS Mode - Selection Flowchart

Main functions include communicating with the stations of assembly, collecting specific product data and assessing the quality of the current product.

3.7. CONCLUSION

The main aim of this Chapter was to show the decisions taken in the development of the QMS. The decisions are taken with the case study in mind, as well as the literature reviewed in the previous Chapter. Research revolved around data management techniques and resulted in the Just-In-Time approach to data handling, as well as data simulation. The QMS adopts a two-pronged approach with Correct data handling, forming the basis of the study and the second, is overcoming the complexities caused by the flexibility of FMSs.

A hierarchical approach to the data handling process, is highlighted as the best approach, with a supervisory element of the system overseeing the flow of the data from the lower levels.

CHAPTER 4: DEVELOPMENT AND DEPLOYMENT OF THE QUALITY MANAGEMENT SYSTEM (QMS)

4.1. INTRODUCTION

Chapter 4 is aimed at highlighting the iQMS functions and the development process. It covers all the aspects discussed in the previous Chapter, which involves the collection and processing of data. Data collection encompasses the collection of training data as well as product data. An initial discussion takes place illustrating the process of data collection. LEARN mode and RUN mode procedures, are presented and discussed. The LEARN mode process is, thereafter, shown from the perspective of the user, with discussions accompanying the illustration. RUN mode is showcased in a similar style. LEARN mode converses on the employment of the QR codes and training data tables. RUN mode, on the other hand, is involved in the product data collection and processing, to qualify the product under development. This is discussed and presented below.

4.2. THE QMS APPLICATION DEPLOYMENT

The QMS is built on a collaborative technology foundation, which involves the interfacing of a C# Software/Control System, a SQL Database and an ARDUINO single-board microcontroller platform. It is an interactive system, designed with ease of use in mind. User interaction is facilitated with prompts that act as a guide, through both the learning and production process of the application. Learning is a function of the system to cater to the reconfiguration of the FMS and, as a result, the change in product data collected and/or processed. The production process refers to the facility that collects and qualifies data for the product undergoing assembly.

Figure 4.1 below, illustrates the initial selection screen of the QMS, with a prompt to SELECT APPLICATION MODE OF OPERATION and a clear selection between LEARN and RUN mode. The third button is used to exit the application and stop all functions.

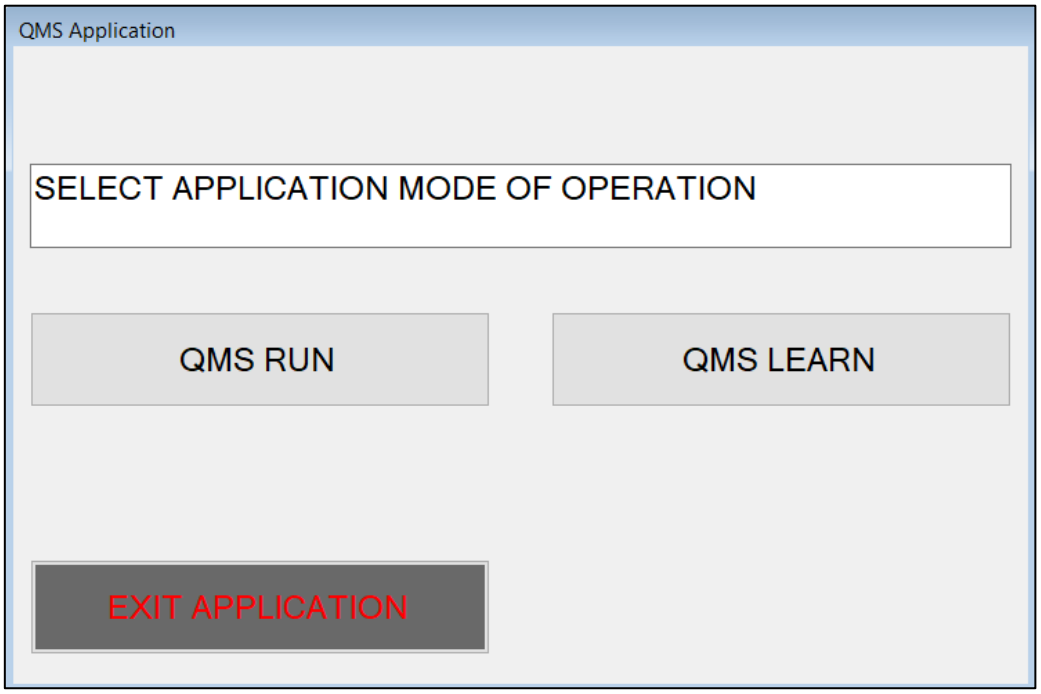


Figure 4.1: QMS - Mode Selection Screen (GUI)

Fulfilling the function of the system, required adding several libraries. The **AForge** library, to access different video sources connected to the PC. The **ZXing** library, an open source, multi-format 1D/2D barcode image processing library. These two libraries are used together, to access the webcam and scan the image, to process the barcode in the frame. The MySQL library was added to the application, to work with the MySQL DB server, as shown in Figure 4.2.

```
using System.Windows.Forms;
using AForge.Video;
using AForge.Video.DirectShow;
using ZXing;
using System.Data.SqlClient;
using MySql.Data.MySqlClient;
```

Figure 4.2: using Reference for The AForge and ZXing NuGet Packages

4.2.1. LEARN Mode

Learning data is compiled in the form of QR codes, selected to represent successful and unsuccessful products. Denoted as FAIL for unsuccessful and PASS for successful, in the product data quality status.

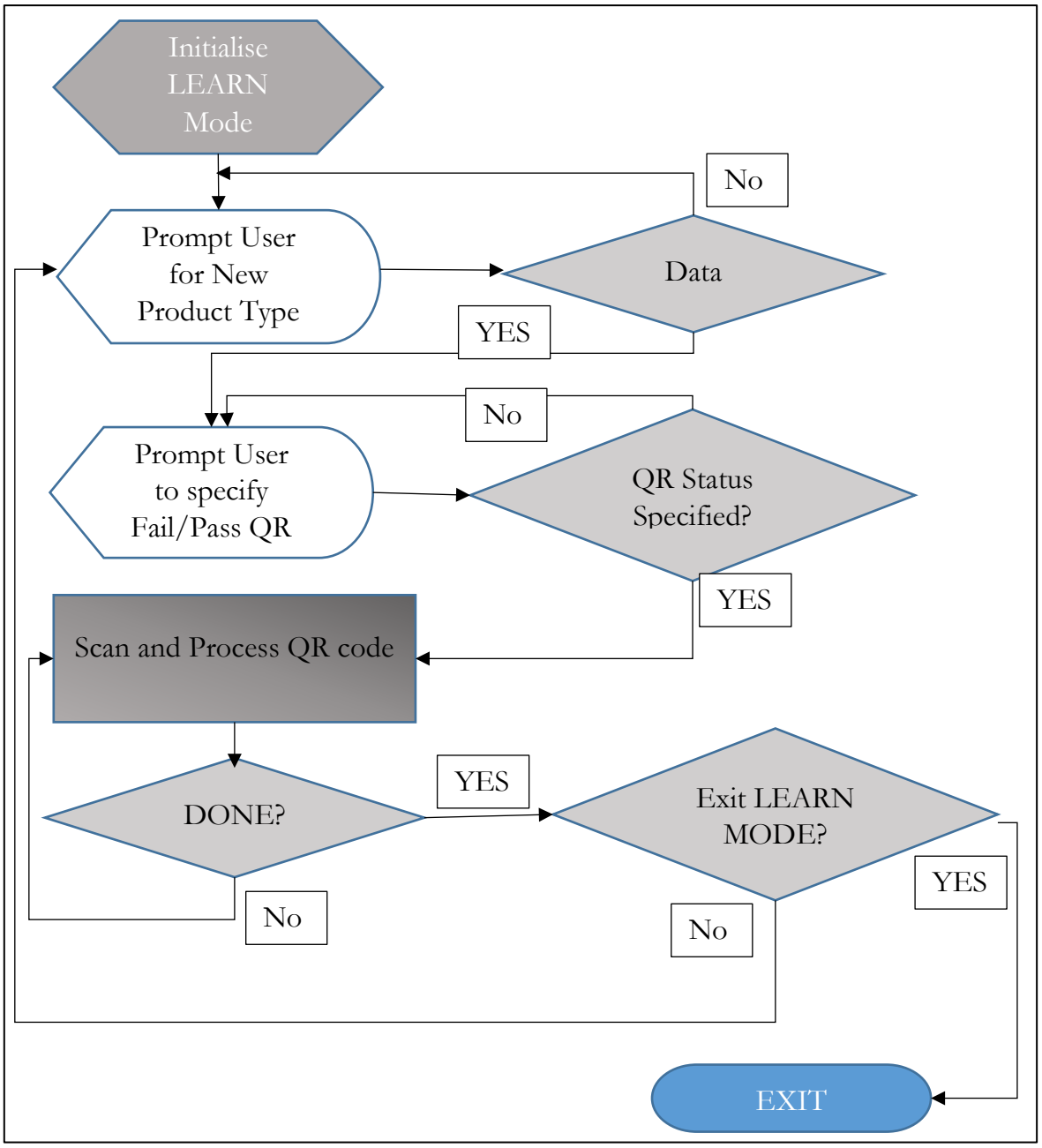


Figure 4.3: LEARN Mode - Flow Chart

Figure 4.3 highlights the main steps involved in the LEARN mode process. The bulk of the work went into verifying that the initial product data was entered by the user and prompting the user to do so, if there was no data. This data is used to update the Product Tracker table of the iqms_app schema, mentioned previously. It is secondarily used to set a table up for the new product, so the Product Type, that the new product belongs to, should be specified, as it will be used to link the new LEARN table to the Product tracker.

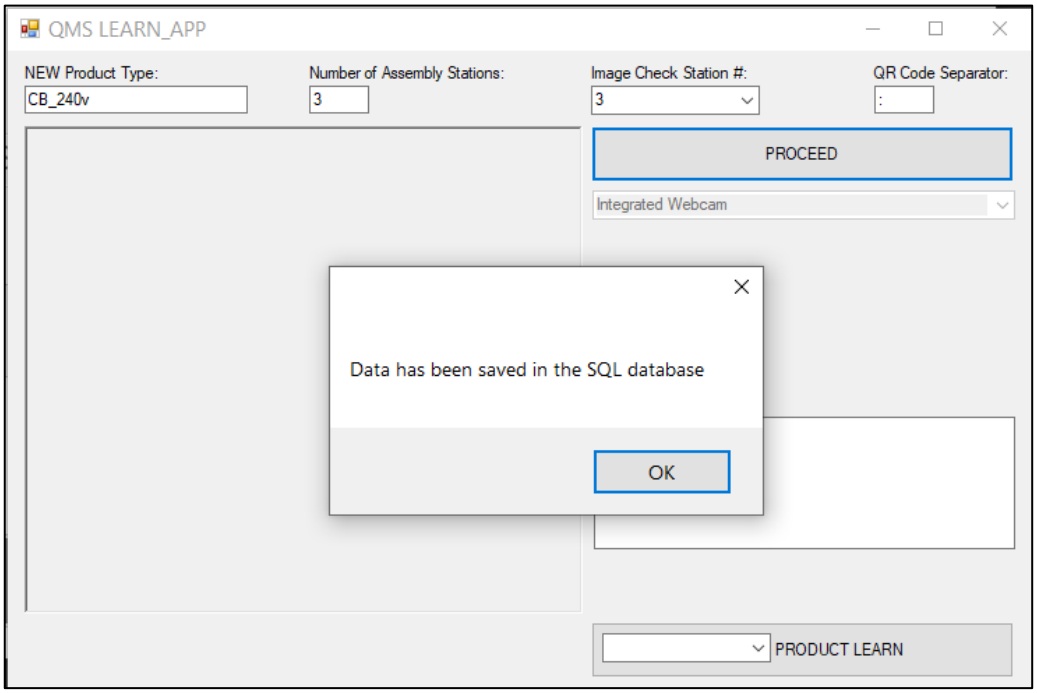


Figure 4.4: Confirmation of NEW Product Type Data Collection

Figure 4.4 illustrates the confirmation of data collection, after product type parameters, as the new product is stored in the Product Tracker table, belonging to the iqms_app schema.

The next step involves selecting the quality of the QR codes that are to be scanned. This should be carried out before scanning the code, which further enables the SCAN CODE button. This is illustrated in Figure 4.5.

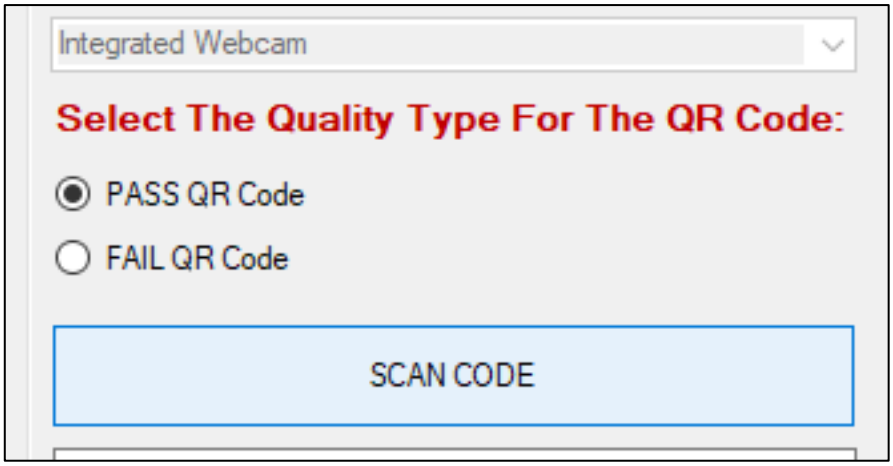


Figure 4.5: Prompt to Categorise the QR Code as Pass or Fail

Scanning and processing of the QR code, takes place thereafter. The application accesses the camera and scans at 1 second intervals, until the QR code is captured and immediately processed to display to the user, as shown in Figure 4.6.

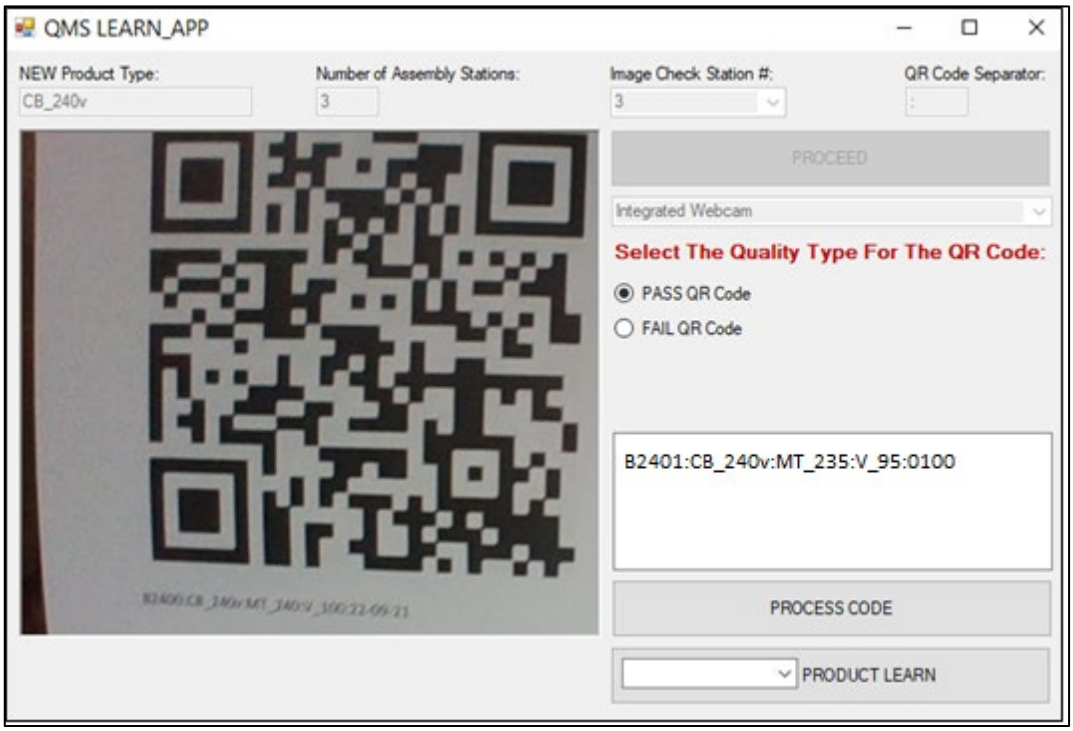
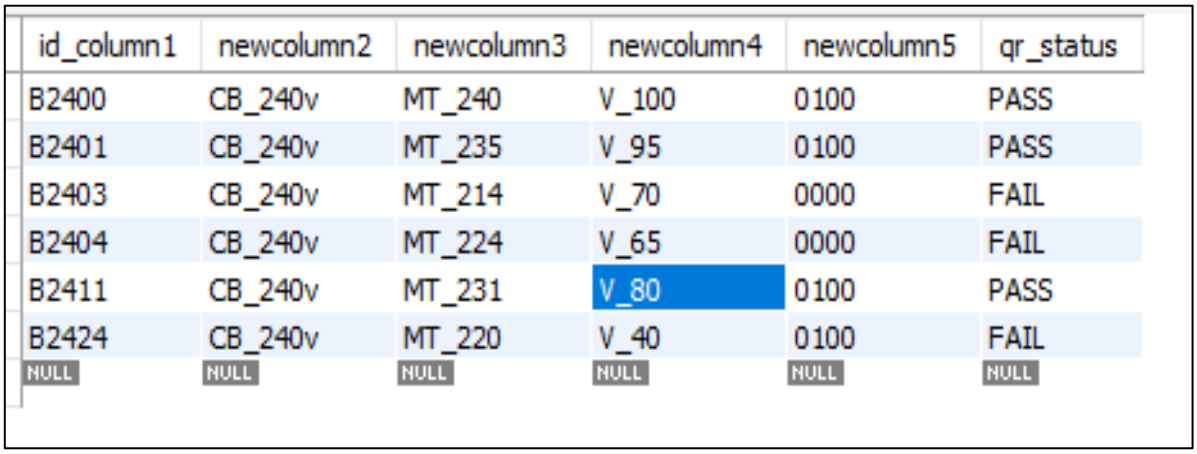


Figure 4.6: New Product Type QR code Processing

Specifying the code separator, results in the QR code data separating, to be stored in columns when added to the new table. The table name is a combination of the Product Type and application mode. The table name of the product scanned above, in Figure 4.6, will be **CB_240v_Learn.**



id_column1	newcolumn2	newcolumn3	newcolumn4	newcolumn5	qr_status
B2400	CB_240v	MT_240	V_100	0100	PASS
B2401	CB_240v	MT_235	V_95	0100	PASS
B2403	CB_240v	MT_214	V_70	0000	FAIL
B2404	CB_240v	MT_224	V_65	0000	FAIL
B2411	CB_240v	MT_231	V_80	0100	PASS
B2424	CB_240v	MT_220	V_40	0100	FAIL
NULL	NULL	NULL	NULL	NULL	NULL

Figure 4.7: Execution of a Select Statement on the CB_240v_Learn Table

Figure 4.7 illustrates the successful implementation of LEARN mode, when viewed using the MySQL Workbench. This was a "**SELECT * FROM iqms_app.cb_240v_learn;**" SQL statement, executed on the Table.

The successful conclusion of the LEARN mode, results in the capturing of data for a particular product. Initially, three product data parameters were introduced to the system. Each product data table generated during the LEARN mode, represents a different arrangement of assembly stations, alerting the system to the new sequence of data to be captured. On Exiting/Closing down the LEARN mode interface, the initial UI is loaded, with the option to go back into LEARN mode or, alternatively, launch RUN mode.

The captured data serves as the training data for the QMS. Consumption of training/learning data increases the correctness of the final product assessment and consistency of the machine learning system/QMS.

4.2.2. RUN Mode

RUN mode is illustrated in Figure 4.8. The initial stage of the operation involves accessing the database and retrieving the last captured product parameters during LEARN mode. Several product data parameters are captured during LEARN mode. This provides the user with the option of selecting alternative product parameters to run.

RUN mode is set up, to process the product data captured, whenever and however, the assembly system is reconfigured. This process is dependent on the learning data captured in the previous process, discussed above.

Highlighted in Figure 4.9 is the LOAD ALTERNATIVE PRODUCTS button, which provides the user with access to the other learned data for products in the database. A single UI is used, to avoid the need for too many prompts and, thus, steps, which may disorientate or lead to common user mistakes.

Following the request to load alternate product data, the user is presented with options in a dropdown list. This list is simplified, displaying solely the circuit breaker type(s) loaded during LEARN mode. Figure 4.10 illustrates the selection of alternate product parameters. The system loaded the parameters for CB_240v when initialised and, then, the user selected the parameters for CB_120vA. The system will follow this up, with a setup, to work with the selected product data.

The RUN LAST LEARNED ASSEMBLY button is thereafter available and is used to revert back to the CB_240v parameters.

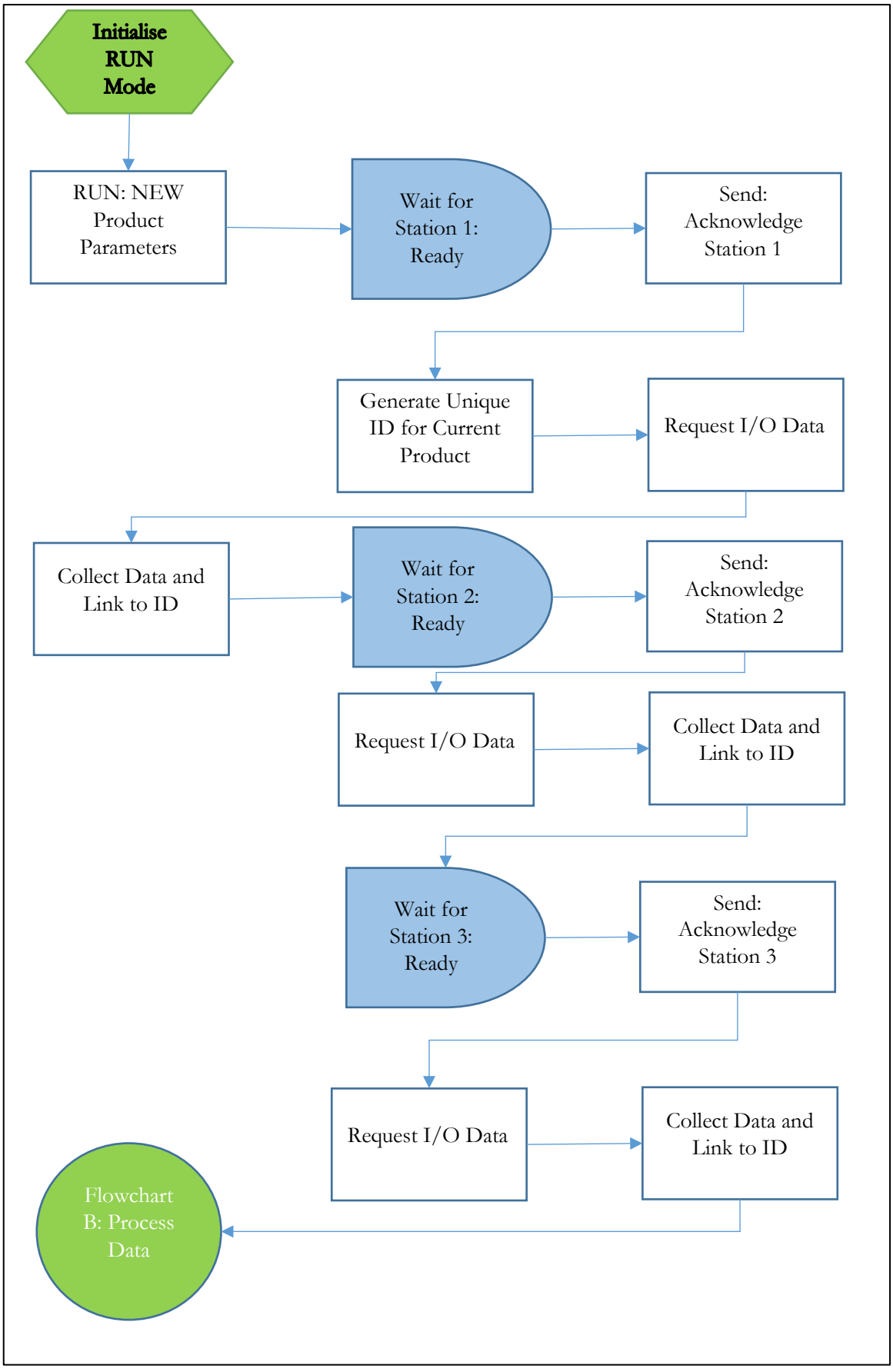


Figure 4.8: RUN Mode - Flowchart (Part A)

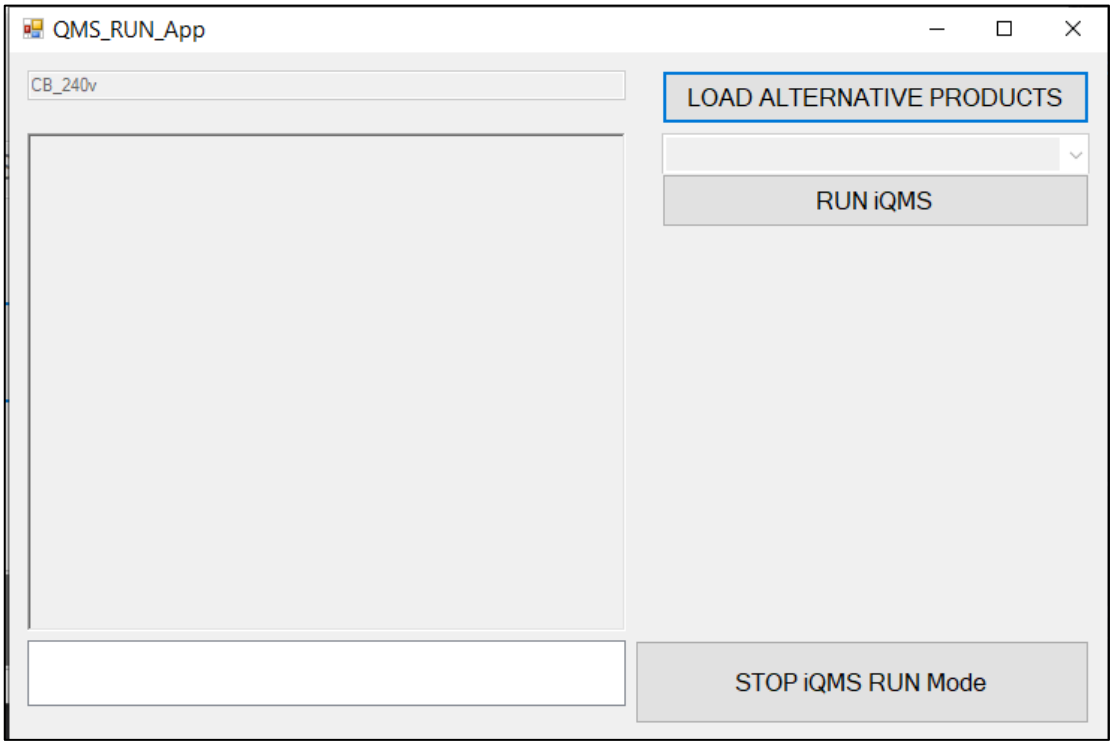


Figure 4.9: RUN Mode - User Interface

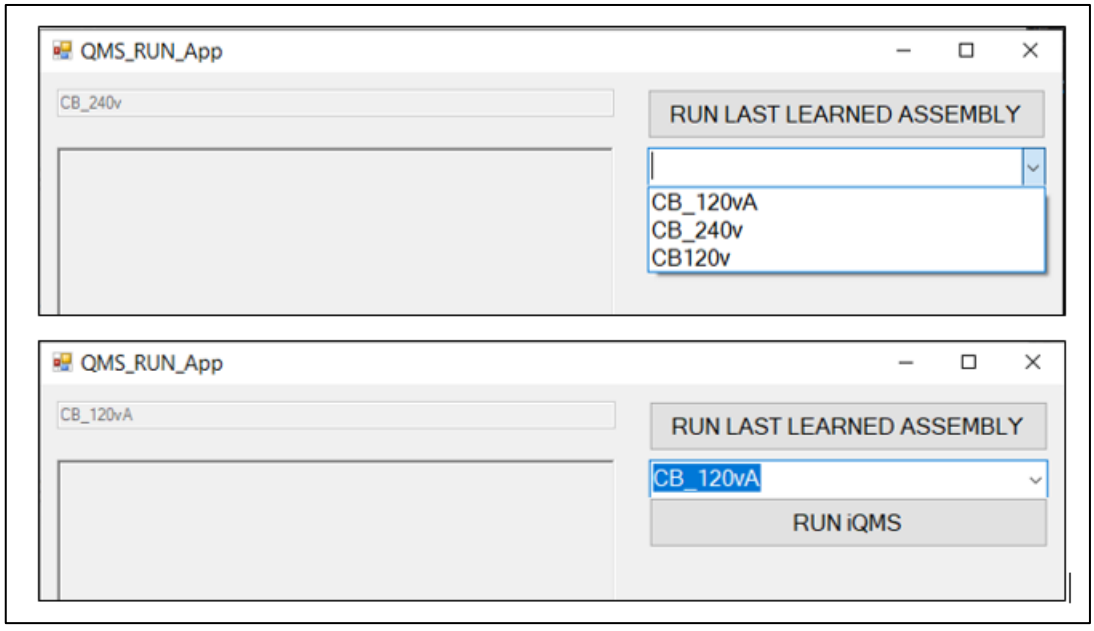


Figure 4.10: RUN Mode - Alternative Product Parameter Selection

Launch/Commencement of RUN mode prepares the system to receive and evaluate product data as it is transmitted from the stations. Communication and transmission is controlled by a series of commands between the hardware nodes; wired to the stations and the QMS. See Table 4.2, a discussion follows.

Table 4.1: Data Communication Commands

Command	**Meaning**	**Comment**
RDY	Ready	Station data is ready to be sent
STBY	Standby	QMS is standing by to receive
Analog *	Analog data	Data being sent is Analog data (*followed by the data)
Digital *	Digital data	Data being sent is Digital data (*followed by the data)

Communication is initiated by a **RDY,** indicating that the station data is *ready* to be sent to the QMS. The QMS sends a **STBY** command and awaits the data from the station. Data is sent with an indication of the type of data it is. The command **Analog** is sent, along with the data to the QMS. This identifies the analogue data transmitted. **Digital** data is sent in the same manner.

Figure 4.11, below, illustrates the QMS in STBY mode. Data is to be transmitted from the hardware nodes. The QMS is awaiting the data.

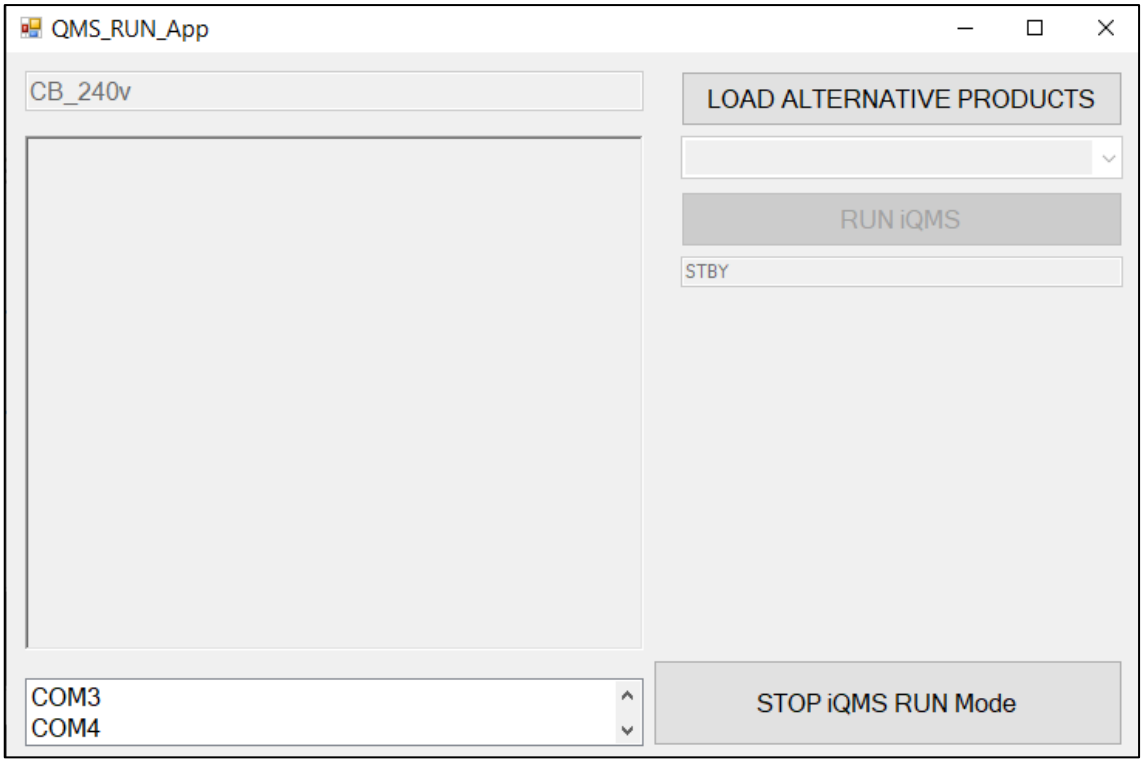


Figure 4.11: RUN Mode Initiated

The **STOP iQMS RUN Mode** button, may be used to interrupt the operation of the QMS, at any point before final evaluation of a product. This will result in the process having to restart again, upon resolution of the interruption. An interruption hinges on the user. Interruptions may be any action that the user sees to not be in order, such as the wrong product parameters selected upon initialisation.

Figure 4.12 illustrates the receiving of Digital data **0000**. This precedes the data storage and evaluation of the received data.

All data received from the stations, is assigned to a unique ID. The ID is generated by retrieving product ID data provided by the learned data and constructing a similar new unique ID. The ID is generated solely when the initial data, from the first station is received. Subsequent ID's are generated, by retrieving data from the newly created tables. These tables are designated **_run,** to reflect that they were created during the RUN mode. Recall the SELECT statement that was constructed to access **cb_240v_learn** table, which was created during LEARN mode. Table **cb_240v_run,** refers to a table created during the

RUN mode and references the aforementioned learn data table. Table 4.3 shows the created product type tables.

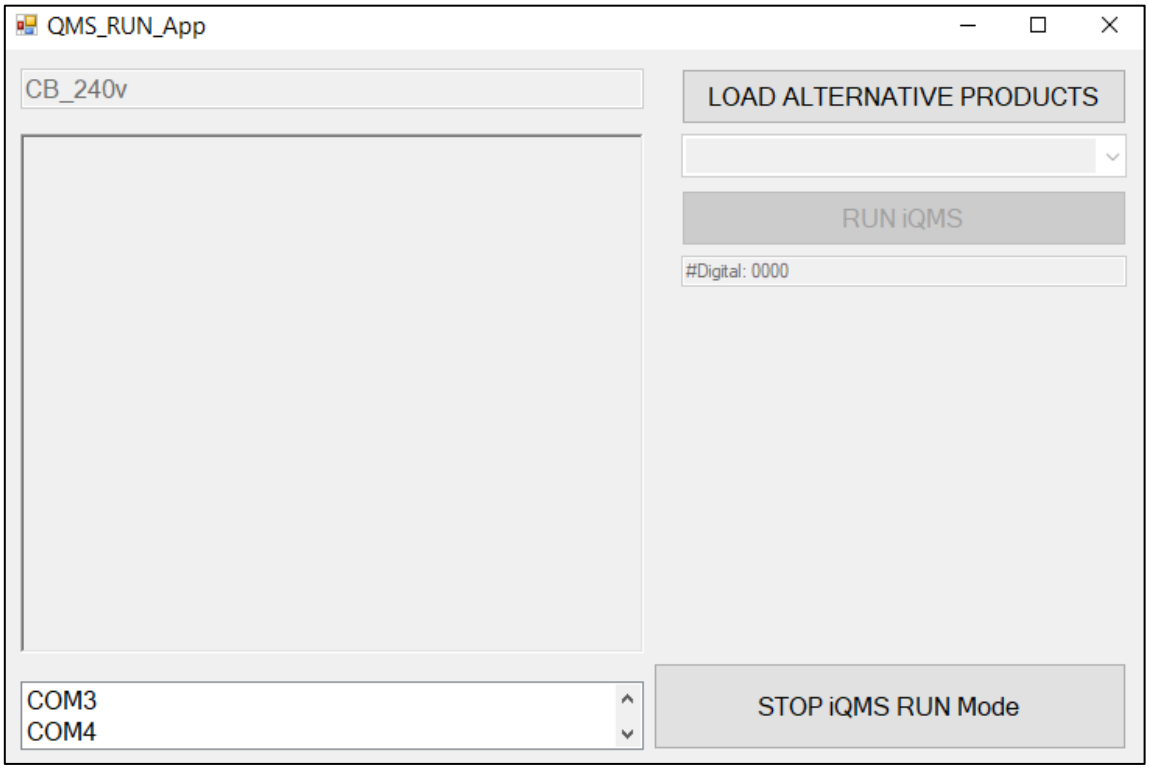


Figure 4.12: RUN Mode Showing Received Digital Data

Table 4.2: Table Names as Created in Different QMS Modes

CB Type	**LEARN Mode Table(s)**	**RUN Mode Table(s)**
CB120v	cb120v_learn	cb120v_run
CB_240v	cb_240v_learn	cb_240v_run
CB_120vA	cb_120va_learn	cb_120va_run

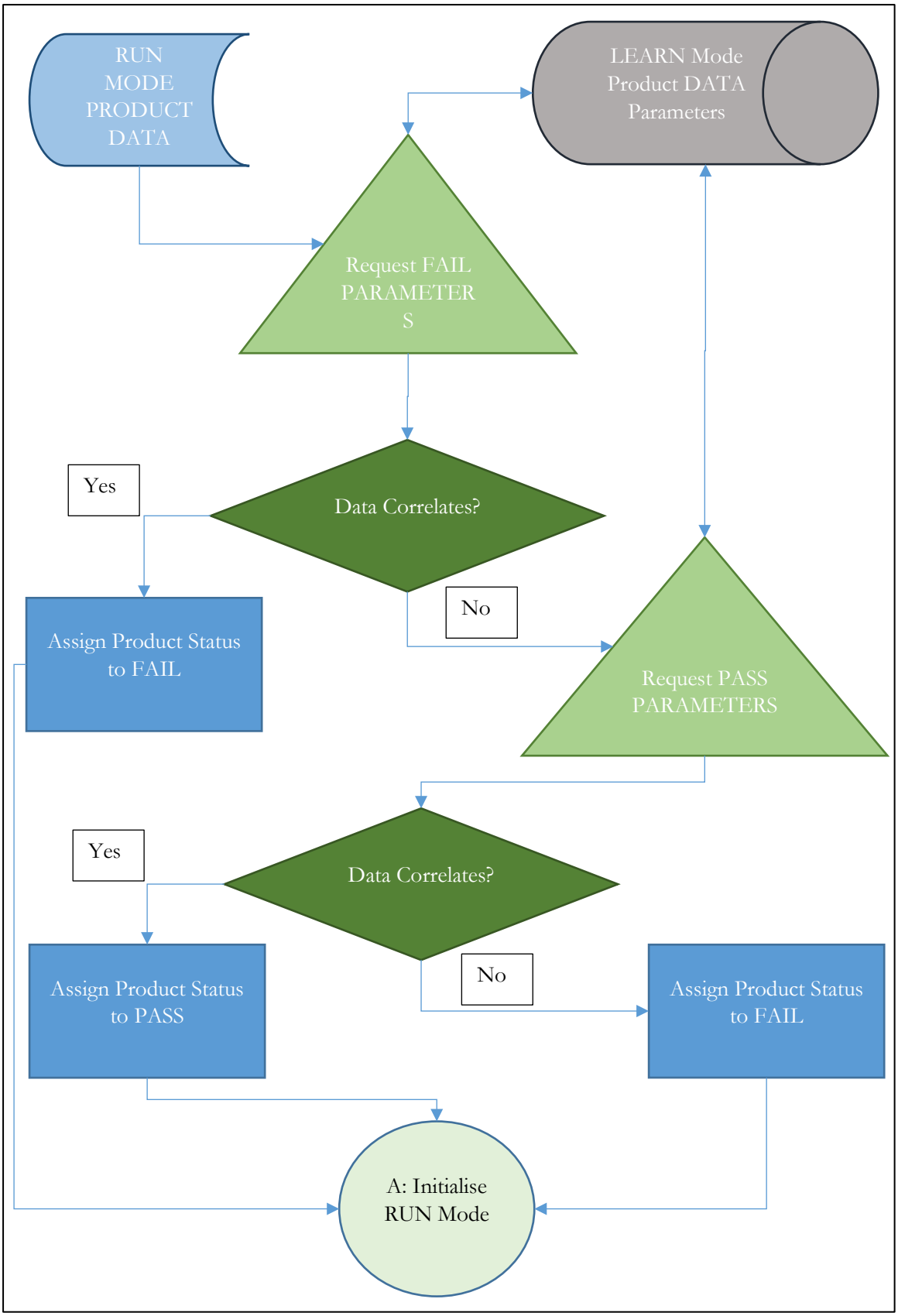


Figure 4.13: RUN Mode: Data Process Cycle (Part B)

The Data Process Cycle, in Figure 4.13 above, illustrates the steps involved in the handling of data subsequent to the steps in Figure 4.8. RUN mode data is linked to the unique ID generated, stored in the database with this ID as the Primary key. The three data variables are, thereafter, used to work out the assignment of the quality status (**qr_status**).

FAIL data parameters are requested from the saved LEARN Mode table, that corresponds to the current product type under assembly. The current captured data variables are compared to the requested data and a positive result means that the current product falls within the failed product parameters, thus, the **qr_status** will be FAIL.

PASS data parameters are requested when the product fails to meet the criteria of the FAIL status. Current captured data variables are compared to the requested learned data. A positive outcome of the comparison, results in the assignment of a PASS to the product **qr_status**.

A product that does not qualify as FAIL or PASS is set to FAIL. The quality of the product is of importance. Assigning a FAIL instead of a PASS, after a failed comparison of both requested parameters, sets an intention to assure a high standard of product quality. The user may take note of this and include more LEARN data for the subsequent product learning process.

4.3. RESULTS

4.3.1. Conversion of QR code to training data

QR code data was compiled from previous product data, which is qualified as either a “good” or “bad” product. “Good”, referring to a product that meets all the requirements of a quality product. “Bad”, referring to a product that does not meet the requirements. Successful scanning and conversion of QR codes eased the training data collection process. User input is further required but, solely to indicate whether the set of QR codes fall into the PASS or FAIL category.

4.3.2. Training Data Table(s) Setup

Training Data Table(s) Setup refers to the successful transfer of the collected data from the QR codes to the database server, creating training data tables. LEARN Mode tables may be seen in Figure 4.14, below.

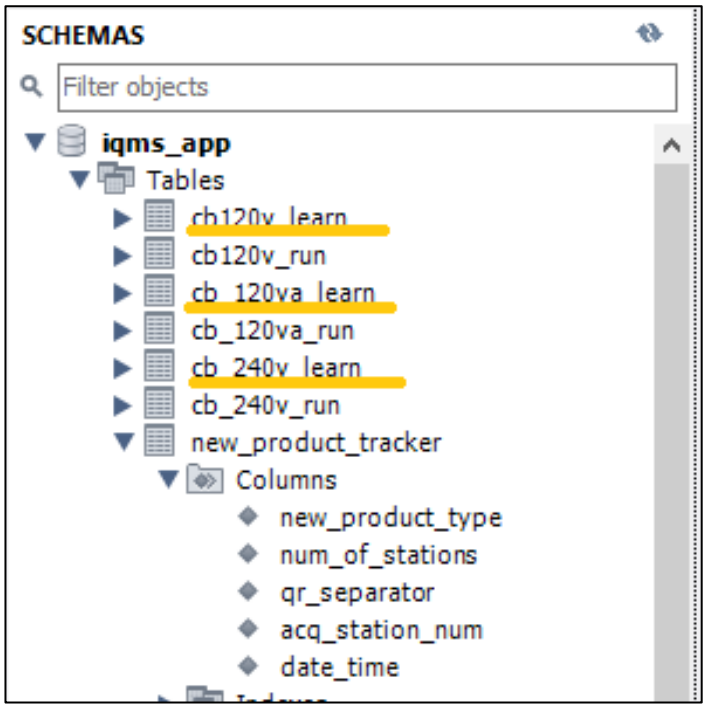


Figure 4.14: LEARN Mode - Training Data Tables

These training data tables, are used to assess the respective product data tables. Assessment of product data, results in the qualification of products as FAIL or PASS. Figure 4.15 illustrates an assessed product data table.

4.3.3. Capturing of Data from Hardware Nodes

RUN Mode product data was successfully captured through the Hardware Nodes. This may be seen in the RUN Mode tables, shown in Figure 4.21, above. The resulting data for **cb_240v_run** table, can be seen below, in Figure 4.15.

prod_id	product_type	newcolumn3	newcolumn4	newcolumn5	qr_status	product_date
B2425	CB_240v	236.99	66.9	0000	FAIL	2021/11/09 3:31:20 PM
B2426	CB_240v	238.99	80.4	0100	PASS	2021/11/09 3:33:19 PM
B2427	CB_240v	233.29	80.4	0100	PASS	2021/11/09 3:33:56 PM
B2428	CB_240v	231.98	91.5	0100	PASS	2021/11/09 3:34:21 PM
B2429	CB_240v	231.80	93.2	0100	PASS	2021/11/09 3:35:34 PM
B2430	CB_240v	231.98	96.5	0101	FAIL	2021/11/09 3:36:14 PM
B2431	CB_240v	230.14	89.2	0100	FAIL	2021/11/09 3:36:43 PM
B2432	CB_240v	236.36	90.3	0100	PASS	2021/11/09 3:40:24 PM
B2433	CB_240v	220.13	89.0	0100	FAIL	2021/11/09 3:41:56 PM
B2434	CB_240v	233.47	90.3	0100	PASS	2021/11/09 3:42:16 PM
B2435	CB_240v	220.48	90.3	0000	FAIL	2021/11/09 3:43:33 PM
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 4.15: RUN Mode - Product CB_240V Data Table

4.3.4. Data Evaluation and Quality Status Assignment

Captured product data is evaluated, based on the data provided during LEARN Mode. The table, as shown in Figure 4.15, references LEARN Mode data from table **cb_240v_learn**. The quality status is assigned when the parameters are matched to the training data referenced. A PASS or FAIL, as shown, is the status assigned.

4.4. GENERATING A REPORT

A prompt report system is included in the project, providing the user with a view of the table data gathered during RUN mode. The report shows the product as designated by product type, therefore, the user should specify which product type they choose to view. Initialisation of the report generator window loads all the product types from the database. The user is tasked with the selection of the product ID and the initiation of the specified product report, by using the GENERATE REPORT button.

The report Generator uses this three-step selection process to query the database and requires the user to guide it, for the sake of accuracy, as well as to prevent a large data

transfer, which may result in the system slowing down. Large data transfers are not catered for, but rather, specific data selection queries are constructed in aid of the user.

Figure 4.16, below, illustrates the user selection of the Product Type. This step initiates a request to the database for all the product ID's that relate to the specified product type.

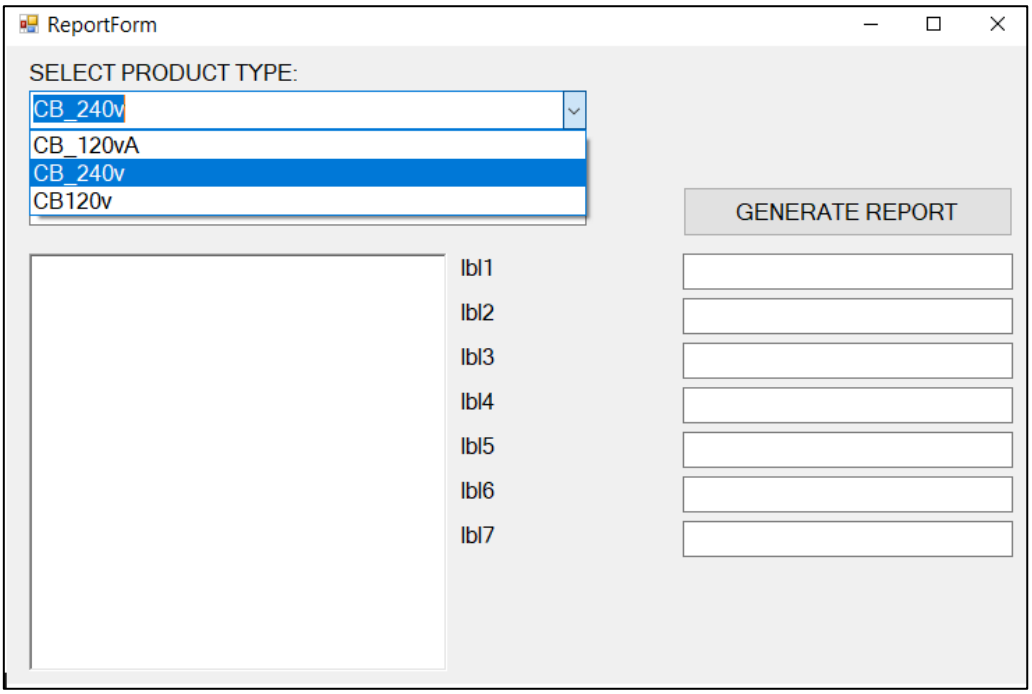


Figure 4.16: Report Generator Form

The following step is the Product ID selection, which is illustrated below in Figure 4.17.

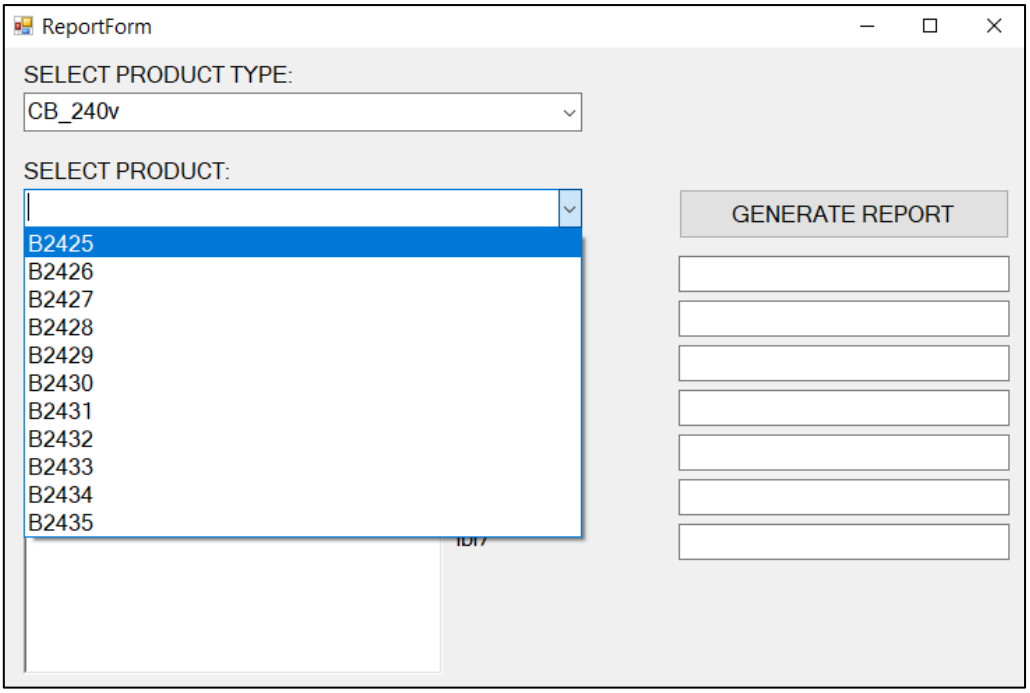


Figure 4.17: Report Generator - Product ID Selection

The GENERATE REPORT button, retrieves all of the product details and populates the window. This further initiates a QR code generation for the selected product, as seen in Figure 4.18.

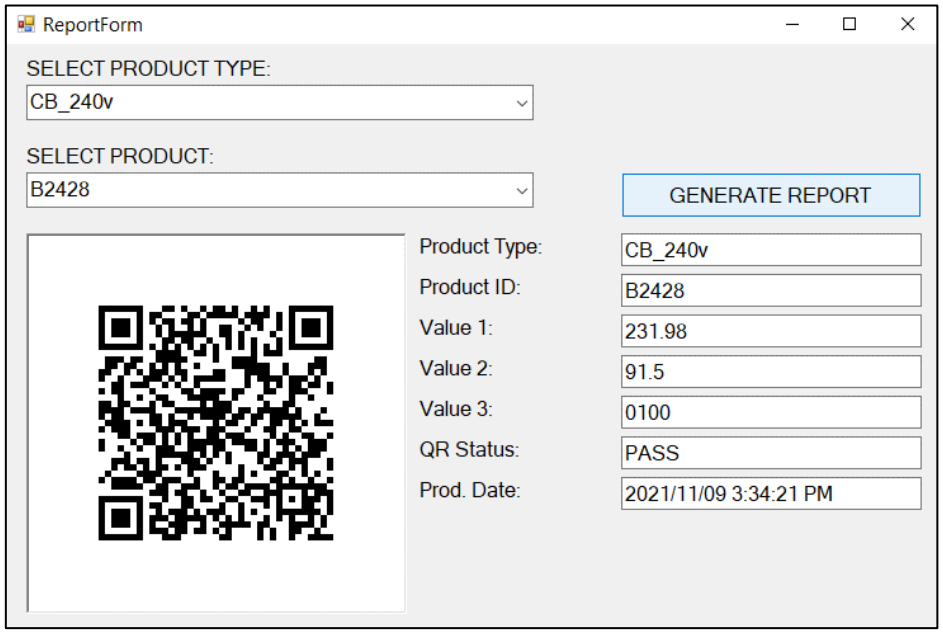


Figure 4.18: Report Generator - Product Details

Data transfers were facilitated through the local instance of the MySQL database facility. SELECT query statements rely on the table construction and fields specified at creation, this knowledge enables the correct syntax of a SELECT statement. The data may be concentrated, by using FROM and WHERE clauses, to filter out other fields.

4.5. CONCLUSION

This Chapter revolved around the development of an innovative quality management system, for a flexible manufacturing system. The aim is to handle data and data changes, as the manufacturing system reconfigures. System reconfiguration, results in data shift and data format changes in the manufacturing system. A station that provided analog data, may be replaced with another that provides digital data and the interfaced data processing unit, may thus misinterpret the data. The QMS is developed with this knowledge of data shifts, as the system reconfigures, to meet product demand and changes. At the start of the study, the primary problem to be resolved, was that of acquiring an innovative method of reprogramming the QMS, in order to handle the new product data.

A case study of circuit breaker assembly was introduced, to highlight the reconfiguration process, as well as the data shift, as reconfiguration takes place. This data reconfiguration is a result of assembly/test stations, that are relocated or replaced to accommodate new products or changes in products. Reconfiguration is an advantage in the manufacturing industry, however as noted, is not exempt from disadvantages. The focus in the case study was test stations. Test stations provide data relating directly to the quality of a product. Quality products meet all the requirements; this serves as proof of their accurate assembly. Additional data is required for all quality products, and this is where testing is involved. Test data stations are under scrutiny throughout the study and training data, gathered during the LEARN process, relates to this.

The innovative method, employed by the QMS, is the use of QR codes to accelerate the training data acquisition process during system LEARN mode. QR code capture and decode is immediate. The subsequent steps to convert to training data that may be stored

in the database, are further relatively quick. Training data is thereafter compiled and ready for the new product parameters introduced to the system.

The resultant data stored during the RUN mode, may be evaluated, using these new product parameters to conclude on the product quality. A PASS or FAIL is assigned as proof of the product correctness, not only in construction but also in function.

The result of the study is an iQMS, that satisfies the data collection and processing needs of the FMS, through the innovative approach employed. The iQMS meets the criteria, as set out in the aim of the study. Ease of reprogramming, with the aid of QR codes, as well as smart use of the database system functionalities to generate training data tables that aid in qualifying the product data captured during assembly.

CHAPTER 5: CONCLUSION AND RESEARCH SUGGESTIONS

5.1. SUMMARY

This Chapter serves as a conclusion of the study, on the resolution of the data management needs of flexible manufacturing. The study steered the production of an information management system specifically matched to the flexible nature of FMSs.

Chapter 1 presented the problem associated with FMSs. The requirements for the solution, were the foundational understanding of the flexibility or reconfigurability of the system. Changes in assembly station positions, resulted in data changes. The data changes meant data format change and possible data handling change. Data capturing (input) and processing (output) should match the change and the reconfiguration of the system. Data interfaces to the information management system should accommodate the quick transition that takes place when the system reconfigures. Furthermore, this also means a type of reconfiguration for the information management system, as data formats and the stations that send the data, have changed.

A discussion in Chapter 2, expanded on traditional FMS and QMS. Literature revealed that the bulk of research was concerned with system optimization and manufacturing industry financial improvements. This largely dealt with system productivity and increasing the operational speed of the FMS. Further research areas revolved around management of tools and parts. The shortfall of traditional QMSs, was discussed, highlighting the focus of the study. The research was the foundation of the intelligent QMS.

In Chapter 3, a case study was introduced, to set the stage for the approach that was taken. The case study directed the research solution decisions. Data collection and analysis methods were formulated; a contribution to the innovative aspect of the QMS.

Chapter 4 was the focal point of the study. The development of the iQMS is presented, highlighting the intelligent aspects of the system. Systems functionality is

illustrated, as well as user interaction, data collection and processing, at the different modes of operation.

The iQMS functions and responses to user interaction, are explored. Simple user interface design (UID) was aimed at anticipating what the user may require navigating the system with ease. Data collection prompts to the user, are illustrated and discussed. Verification of data collection and processing, communicates the progress of the system to the user. The system operation modes are similarly designed, with slight changes, to facilitate the collection of training data and the evaluation of manufactured products.

The amount of LEARN data, further known as training data, expands the information that may be used to qualify the product data gathered when running the assembly process. This simply means that, the correctness of successful data processing, increases in proportion to the amount of training data captured. The resulting training data tables may be loaded with as many data points as needed, to increase accuracy. The innovation employed to respond to the reconfiguration of the system, is the QR code aided reprogram. The reprogram is facilitated by the processing of QR code data. QR codes are a convenient data input. They may be compiled from existing product data and may be as data dense as possible, whilst still occupying a specified space. A quick scan and decode, reveals the product data parameters, to be used as LEARN data.

Product data tables were assessed with the use of training data tables. The result of the product evaluation (PASS or FAIL), for the different parameters gathered during product data collection, was proof of system design and development. Successful QR code data processing, generates training data tables, that correlate product data. This data correlation assesses the assembled product, using learned product parameters. Product data parameters are used for the new product type, as many iterations as the system produces. Reports on the product specifications are available to the user, to be generated as necessary.

5.2. CONTRIBUTION TO THE RESEARCH

The main contributions of the study, in comparison with further available research, regarding the manufacturing industry and industry 4.0, are the following:

- An innovative and intelligent data management process, that caters to the flexibility and unique data management requirements of Flexible Manufacturing Systems.
- A quality management research study, in the era of Industry 4.0, highlighting as to how quality management practices may adapt to new digital advances, in the manufacturing industry.

The contributions of the study are driven by the need for more quality data management research in the flexible manufacturing industry. Research that may be scaled to a FMS, that is responsible for the assembly and quality validation of a variable product. The Circuit Breaker case study is a valuable instrument in this regard. The CB is a complex product and its application in the protection of an electrical system, is vital. The QMS study is a rendition of a data management system, uniquely tasked to the assurance of the system accuracy.

5.3. SUGGESTIONS FOR FUTURE RESEARCH STUDIES

Future research may be carried out, with the following as a guide:

- Flexible Manufacturing is vital when it comes to Intelligent Manufacturing and forms the foundation for Digital Manufacturing. Optimization of existing FMS's is thus imperative. Design and development of Control Systems should include a facility for self-analysis of the FMS, to optimise the system.
- Improvements on Quality management should be in proportion to improvements in manufacturing systems.
- Company data security ensures an advantage over competitors. Secure user access may be prioritised, when developing data handling systems.

- The development of a single plug-in connector, for peripheral interfacing, to both digital and analogue signals.

BIBLIOGRAPHY

[1] S. D. Cataldo, S. Lee, E. Macii and B. Vogel-Heuser, "Leading Information and Communication Technologies for Smart Manufacturing: Facing the New Challenges and Opportunities of the 4th Industrial Revolution," *Proceedings of the IEEE, Volume 109,* pp. 320-325, April 2021.

[2] M. R. Pedersen, L. Nalpantidis, R. S. Andersen, C. Schou, S. Bøgh, V. Krüger and O. Madsen, "Robot skills for manufacturing: From concept to industrial deployment," *Robotics and Computer-Integrated Manufacturing,* pp. 282-291, February 2016.

[3] B. Ślusarczyk, "INDUSTRY 4.0 – ARE WE READY?," *Polish Journal of Management Studies; Częstochowa,* pp. 232-248, 2018.

[4] I. P. Gania, A. Stachowiak and J. Oleśków-Szłapka, "FLEXIBLE MANUFACTURING SYSTEMS: INDUSTRY 4.0 SOLUTION," *idea, Volume 10,* pp. 7-11, 2017.

[5] Y. Liu, A. Soroka, L. Han, J. Jian and M. Tang, "Cloud-based big data analytics for customer insight-driven design innovation in SMEs," *International Journal of Information Management, Volume 51,* p. 102034, April 2020.

[6] A. Issa, B. Hatiboglu, A. Bildstein and T. Bauerhansl, "Industrie 4.0 roadmap: Framework for digital transformation based on the concepts of capability maturity and alignment," *Procedia CIRP, Volume 72,* pp. 973-978, 27 June 2018.

[7] M. A. Youssef and B. Al-Ahmady, "The impact of using flexible manufacturing systems on quality management practices," *Total Quality Management,* pp. 813-825, 25 August 2010.

[8] S. Jovanović, "Flexible Manufacturing Systems and Quantitative Anlysis of Flexible Manufacturing Systems," *International Journal of Computer Applications,* pp. 6-14, 2015.

[9] A. Florescu and S. A. Barabas, "Modeling and Simulation of a Flexible Manufacturing System—A Basic Component of Industry 4.0," *Applied Sciences,* p. 8300, 19 November 2020.

[10] K. A. Pupkov and Y. K. Brovarskaya, "Dynamic and Information Properties of Intelligent Control Systems," *Procedia Computer Science,* pp. 489-494, 11 June 2021.

[11] S. Phuyal, D. Bista and R. Bista, "Challenges, opportunities and future directions of smart manufacturing: A State of Art review," *Sustainable Futures 2,* p. 100023, 1 January 2020.

[12] G. Kaur, "OPERATIONAL PROBLEMS IN FLEXIBLE MANUFACTURING SYSTEMS: A REVIEW," *All in one, Volume 2,* p. 6, 2017.

[13] W. Shen and D. Norrie, "Agent-Based Systems for Intelligent Manufacturing: A State-of-the-Art Survey," *Knowledge and Information Systems,* vol. 1, pp. 129-156, 1999.

[14] M. Behun, B. Gavurova, A. Tkacova and A. Kotaskova, "The impact of the manufacturing industry on the economic cycle of European Union countries," *Journal of competitiveness 10,* pp. 23-39, 2018.

[15] A. Petrillo, F. De Felice, R. Cioffi and F. Zomparelli, "Fourth industrial revolution: Current practices, challenges, and opportunities," *Digital transformation in smart manufacturing,* pp. 1-20, February 2018.

[16] K. Schwab, "The Fourth Industrial Revolution: what it means, how to respond," The World Economic Forum, 14 January 2016. [Online]. Available: https://www.weforum.org/agenda/2016/01/the-fourth-industrial-revolution-what-it-means-and-how-to-respond/. [Accessed 20 September 2019].

[17] F. Yang and S. Gu, "Industry 4.0, a revolution that requires technology and national strategies," *Complex & Intelligent Systems 7,* pp. 1311-1325, 27 January 2021.

[18] A. G. Frank, L. S. Dalenogare and N. F. Ayala, "Industry 4.0 technologies: Implementation patterns in manufacturing companies," *International Journal of Production Economics, Volume 210,* pp. 15-26, April 2019.

[19] D. T. Pham and E. Oztemel, Intelligent Quality Systems, London: Springer-Verlag, 1996.

[20] R. C. Luo and M. G. Kay, Multisensor Integration and Fusion for Intelligent Machines and Systems, Norwood: Ablex Publishing Corporation, 1995.

[21] B.-h. Li, B.-c. Hou, W.-t. Yu, X.-b. Lu and C.-w. Yang, "Applications of artificial intelligence in intelligent manufacturing: A review," *Frontiers of Information Technology & Electronic Engineering,* pp. 86-96, 4 February 2017.

[22] J. Davis, T. Edgar, J. Porter, J. Bernaden and M. Sarli, "Smart manufacturing, manufacturing intelligence and demand-dynamic performance," *Computers & Chemical Engineering, Volume 47,* pp. 145-156, 20 December 2012.

[23] R. Y. Zhong, X. Xu, E. Klotz and S. T. Newman, "Intelligent Manufacturing in the Context of Industry 4.0: A Review," *Engeneering,* pp. 616-630, October 2017.

[24] Y. Liu and X. Xu, "Industry 4.0 and Cloud Manufacturing: A Comparative Analysis," *Journal of Manufacturing Science and Engineering, Volume 139,* pp. 1-8, March 2017.

[25] M. Helu and T. Hedberg Jr., "Enabling Smart Manufacturing Research and Development using a Product Lifecycle Test Bed," *Procedia Manufacturing,* pp. 86-97, 21 OCtober 2015.

[26] H. W. M. Zijm, "Towards intelligent manufacturing planning and control systems," *OR-Spektrum 22,* pp. 313-345, August 2000.

[27] W. Shen, Q. Hao, S. Wang, Y. Li and H. Ghenniwa, "An agent-based service-oriented integration architecture for collaborative intelligent manufacturing," *Robotics and Computer-Integrated Manufacturing, Volume 23,* pp. 315-325, June 2007.

[28] W. Shen, Q. Hao, H. J. Yoon and D. H. Norrie, "Applications of agent-based systems in intelligent manufacturing: An updated review," *Advanced Engineering Informatics, Volume 20,* pp. 415-431, October 2006.

[29] G. Huang, Y. Zhang and P. Jiang, "RFID-based wireless manufacturing for real-time management of job shop WIP inventories," *The International Journal of Advanced Manufacturing Technology,* pp. 752-764, 19 January 2007.

[30] M. Wang, R. Y. Zhong, Q. Dai and G. Q. Huang, "A MPN-based scheduling model for IoT-enabled hybrid flow shop manufacturing," *Advanced Engineering Informatics, Volume 30,* pp. 728-736, October 2016.

[31] Y. Zhang, P. Jiang and G. Huang, "RFID-based smart Kanbans for Just-In-Time manufacturing," *International Journal of Materials and Product Technology, Volume 33,* pp. 170-184, 31 July 2008.

[32] R. Y. Zhong, Y. Peng, F. Xue, J. Fang, W. Zou, H. Luo, T. S. Ng, W. Lu, G. Q. P. Shen and G. Q. Huang, "Prefabricated construction enabled by the Internet-of-Things," *Automation in Construction, Volume 76,* pp. 59-70, April 2017.

[33] T. Qu, Z. Z. Wang, D. X. Nie, X. Chen and G. Huang, "IoT-based real-time production logistics synchronization system under smart cloud manufacturing," *The International Journal of Advanced Manufacturing Technology 84,* pp. 147-164, 10 May 2015.

[34] B. Huang, C. Li and F. Tao, "A chaos control optimal algorithm for QoS-based service composition selection in cloud manufacturing system," *Enterprise Information Systems, Volume 8,* pp. 445-463, 2014.

[35] A. B. Feeney, S. P. Frechette and V. Srinivasan, "A Portrait of an ISO STEP Tolerancing Standard as an Enabler of Smart Manufacturing Systems," *Journal of Computing and Information Science in Engineering (JCISE),* 1 June 2015.

[36] FESTO, "Festo Group: Qualification for Industry 4.0," 15 March 2016. [Online]. Available:

https://www.festo.com/net/SupportPortal/Files/427050/brochure_Qi4_screen_56759en.pdf. [Accessed September 2017].

[37] A. Fundin, B. Bergquist, H. Eriksson and I. Gremyr, "Challenges and propositions for research in quality management," *International Journal of Production Economics, Volume 199,* pp. 125-137, May 2018.

[38] D. Hoyle, ISO 9000: Quality Systems Handbook,, London: Routledge, 2006.

[39] D. Li, Y. Zhao, L. Zhang, X. Chen and C. Cao, "Impact of quality management on green innovation," *Journal of Cleaner Production,* pp. 462-470, 1 January 2018.

[40] J. Angelva and P. Piltonen, "Real-time data management in a flexible manufacturing system (FMS)," *Journal of Materials Processing Technology, Volume 52,* pp. 76-82, May 1995.

[41] U. A. W. Tetzlaff, Optimal Design of Flexible Manufacturing Systems, Springer Science and Businesss Media, 2013.

[42] R. M. da Silva, F. Junqueira and D. J. S. Filho, "Control architecture and design method of reconfigurable manufacturing systems," *Control Engineering Practice, Volume 49,* pp. 87-100, April 2016.

[43] E. Abele, T. Liebeck and A. Wörn, "Measuring Flexibility in Investment Decisions for Manufacturing Systems," *CIRP Annals, Volume 55,* pp. 433-436, 2006.

[44] R. Zurawski, Integration Technologies for Industrial Automated Systems, 1st Edition, Boca Raton: CRC Press, 2006.

[45] P. R. Spena, P. Holzner, E. Rauch, R. Vidoni and D. T. Matt, "Requirements for the Design of flexible and changeable Manufacturing and Assembly Systems: a SME-survey," *Procedia CIRP. Volume 41,* pp. 207-212, 2016.

[46] A. Kapitanov, "Manufacturing System Flexibility Control," *Procedia Engineering, vol. 206,* pp. 1470-1475, 2017.

[47] G. Rosati, M. Faccio, A. Carli and A. Rossi, "Fully flexible assembly systems (F-FAS): a new concept in flexible automation," *Assembly Automation, Volume 33,* pp. 8-21, 15 February 2013.

[48] A. M. Gibb, *New media art, design and the Arduino microcontroller: A malleable tool (Master's Thesis),* 2010.

[49] "What is Arduino? arduino.cc," ARDUINO, 5 February 2018. [Online]. Available: https://www.arduino.cc/en/Guide/Introduction. [Accessed 20 March 2018].

[50] H. Nylund, V. Valjus, V. Toivonen, M. Lanz and H. Neiminen, "The virtual FMS – an engineering education environment," *Procedia Manufacturing, Volume 31,* pp. 251-257, 23 April 2019.

[51] M. Somasundaram, M. Junaid and S. Mangadu, "Artificial Intelligence (AI) Enabled Intelligent Quality Management System (IQMS) For Personalized Learning Path," *Procedia Computer Science, Volume 172,* pp. 438-442, 16 June 2020.

[52] S. Gopi and S. A. Chandra, "Solving distributed FMS scheduling problems with/ without breakdowns: Simulation optimization approach," *Materials Today: Proceedings, Volume 47,* pp. 4879-4884, 8 October 2021.

[53] R. Sanchez-Marquez, J. M. A. Guillem, E. Vicens-Salort and J. J. Vivas, "Diagnosis of quality management systems using data analytics – A case study in the manufacturing sector," *Computers in Industry, Volume 115,* February 2020.

[54] K. Mahmood, T. Karaulova, T. Otto and E. Shevtshenko, "Performance Analysis of a Flexible Manufacturing System (FMS)," *Procedia CIRP,* pp. 424-429, 11 July 2017.

[55] C. J. Testa, C. Hu, K. Shvedova, W. Wu, R. Sayin, F. Casati, B. S. Halkude, P. Hermant, D. E. Shen, A. Ramnath, Q. Su, S. C. Born, B. Takizawa, S. Chattopadhyay, T. F. O'Connor, X. Yang, S. Ramanujam and S. Mascia, "Design and Commercialization of an End-to-End Continuous Pharmaceutical Production Process: A Pilot Plant Case Study," *Organic Process Research & Development (OPR&D), Volume 24,* pp. 2874-2889, 21 October 2020.

[56] G. Lugaresi, V. V. Alba and A. Matta, "Lab-scale Models of Manufacturing Systems for Testing Real-time Simulation and Production Control Technologies," *Journal of Manufacturing Systems,* pp. 93-108, 4 December 2020.

[57] Z. Bi, Y. Jin, P. Maropoulos, W.-J. Zhang and L. Wang, "Internet of things (IoT) and big data analytics (BDA) for digital manufacturing (DM)," *International Journal of Production Research,* pp. 1-18, 30 June 2021.

[58] H. Singh, "Big data, industry 4.0 and cyber-physical systems integration: A smart industry context," *Materials Today: Proceedings,* pp. 157-162, 7 June 2021.

[59] D. Luo, Z. Guan, C. He, Y. Ging and L. Yeu, "Data-driven cloud simulation architecture for automated flexible production lines: application in real smart factories," *International Journal of Production Research,* pp. 1-23, 1 May 2021.

[60] A. Zonnenshain and R. S. Kenett, "Quality 4.0—the challenging future of quality engineering," *uality Engineering, Volume 32,* pp. 614-626, 1 October 2020.

[61] A. V. Carvalho, D. V. Enrique, A. Chouchene and F. Charrua-Santos, "Quality 4.0: An Overview," *Procedia Computer Science, Volume 181,* pp. 341-346, 22 February 2021.

[62] J. Wang, C. Xu, J. Zhang and R. Zhong, "Big data analytics for intelligent manufacturing systems: A review," *Journal of Manufacturing Systems, Volume 62,* pp. 738-752, January 2022.

[63] K. M. Khan, I. Hussain and S. Noor, "2. M. A Knowledge Based Methodology for Planning and Designing of a Flexible Manufacturing System (FMS)," *International Journal of Advanced Manufacturing Systems, Volume 13,* pp. 91-106, 2011.

[64] A. Caggiano, F. Caiazzo and R. Teti, "Digital Factory Approach for Flexible and Efficient Manufacturing Systems in the Aerospace Industry," *Procedia CIRP, Volume 37,* pp. 122-127, 2015.

[65] J. Rybicka, A. Tiwari and S. Enticott, "Rybicka, Justyna, Ashutosh Tiwari, and Shane Enticott. "Testing a flexible manufacturing system facility production capacity through discrete event simulation: automotive case study," *International Journal of Mechanical, Aerospace, Industrial, Mechatronic and Manufacturing Engineering, Volume 10,* pp. 690-694, 2016.

[66] J. Banks, J. S. Carson, B. L. Nelson and D. M. Nicol, Discrete-Event System Simulation, Harlow: Pearson Higher Ed., 2013.

[67] L. Bellomarini, R. R. Fayzrakhmanov, G. Gottlob, A. Kravchenko, E. Laurenza and Y. Nenov, "Data science with Vadalog: Knowledge Graphs with machine learning and reasoning in practice," *Future Generation Computer Systems,* pp. 407-422, 8 November 2021.

[68] N. T. Ching, M. Ghobakhloo, M. Iranmanesh, P. Maroufkhani and S. Asadi, "Industry 4.0 applications for sustainable manufacturing: A systematic literature review and a roadmap to sustainable development," *Journal of Cleaner Production, Volume 334,* pp. 130-133, 1 February 2022.

[69] D. Lichtenwater, P. Burggräf, J. Wagner and T. Weißer, "Deep multimodal learning for manufacturing problem solving," *Procedia CIRP, Volume 99,* pp. 615-620, 3 May 2021.

APPENDICES

Appendix 1: Hardware Node Program using ARDUINO Microcontroller

```
#include <EEPROM.h>
//#include <Servo.h>
#include <Arduino.h>

String inputString = "";         // a string to hold incoming data
boolean stringComplete = false;  // when the string is complete
String commandString = "";

int sensorPin1 = A0;    // select the analog inputs
int sensorPin2 = A1;
int sensorPin3 = A2;
int sensorPin4 = A3;
int sensorPin5 = A4;
int sensorPin6 = A5;

String sensorValue1 = "";  // variable to store the value coming from the sensor
String sensorValue2 = "";
String sensorValue3 = "";
String sensorValue4 = "";
String sensorValue5 = "";
```

```
String sensorValue6 = "";


int Arr_In = 2;       // select the digital inputs: Sensor for Current Assembly Point
int digital_In2 = 3;
int digital_In3 = 4;
int digital_In4 = 5;
//int digital_In5 = 6;
//int digital_In6 = 7;
//int digital_In7 = 6;
//int digital_In8 = 7;


int ST_Out = 8;       // select the digital outputs
int digital_Out2 = 9;
int digital_Out3 = 10;
int digital_Out4 = 11;


const int ledPin =  13;    // select the pin for the Build-in LED


// variables will change:
int buttonState = 0;         // variable for reading the pushbutton status
int LEDState = 0;
int AD_state = 0;


//Other variables for communication with QMS
  int QMS_Alert = LOW;
```

```
//int QMS_FBack = LOW;
//int QMS_Pending = LOW;

void setup()
{
        // initiate serial port
        Serial.begin(9600);
        while (!Serial)
 {
  ; // wait for serial port to connect. Needed for native USB port only
 }
 // declare the ledPin as an OUTPUT:
 pinMode(ledPin, OUTPUT);

 //declare INTPUTs

 // declare OUTPUTs

 }

void loop()
{

// read the state of the pushbutton value:
 buttonState = digitalRead(Arr_In);
```

```
// Stop Conveyor when Product Arrives
if ((buttonState == HIGH)&& (LEDState == LOW))
{
 digitalWrite(ST_Out, HIGH);

 QMS_Alert = HIGH;

 //Alert Main Program (C#)
 if(QMS_Alert == HIGH)
 {
  String stringAlert = "#RDY@";
  while(Serial.available() <= 0)
  {
     Serial.print(stringAlert);

     delay(300);
  }
 }

 //Open Communication

}
else  if ((buttonState == LOW) && (digitalRead(ST_Out) == HIGH))
{
```

```
 digitalWrite(ST_Out, LOW);
 }

if(stringComplete)
{
 stringComplete = false;
 getCommand();

 if(commandString.equals("DATA")) //Request for Digital data
 {
   digitalWrite(ledPin,HIGH);
   AD_state = digitalRead(digital_In4);

   delay(1000);
   //Read Node digital data
   String relay1 = String(digitalRead(Arr_In));
   String relay2 = String(digitalRead(digital_In2));
   String relay3 = String(digitalRead(digital_In3));
   String relay4 = String(digitalRead(digital_In4));

   String relaystatestring = String("#Digital: " + relay1 + relay2 + relay3 + relay4 + "@");

 //Read Node Analog Data
```

```
int sensorValue1 = analogRead(sensorPin1);


float value = sensorValue1 * (100.0/512);


sensorValue2 = String(analogRead(sensorPin2));
sensorValue3 = String(analogRead(sensorPin3));
sensorValue4 = String(analogRead(sensorPin4));
sensorValue5 = String(analogRead(sensorPin5));
sensorValue6 = String(analogRead(sensorPin6));


String ConcatValue = String("#Analog: " + String(value)  + "@"); // + ": " + sensorValue2 + ": " + sensorValue3 + ": " + sensorValue4 + ": " + sensorValue5 + ": " + sensorValue6);


if(AD_state == LOW)
{
//
//print out digital values
Serial.println(relaystatestring);
}
if(AD_state == HIGH)
{
Serial.println(ConcatValue);
}
delay(1000);
```

```
    digitalWrite(ledPin, LOW);


    }
    //Optional Confirmation Commands
    else if(commandString.equals("ACKN"))
    {


      //buttonState = LOW;
      QMS_Alert = LOW;
      Serial.println("#STBY@");
      digitalWrite(ledPin,HIGH);
    }
    inputString = "";
  }
}
//
void serialEvent()
{
  while (Serial.available())
  {
    // get the new byte:
    char inChar = (char)Serial.read();
    // add it to the inputString:
    inputString += inChar;
    // if the incoming character is a newline, set a flag
```

```
    // so the main loop can do something about it:
    if (inChar == '\n')
    {
      stringComplete = true;
    }
  }
}

//
void getCommand()
{
  if(inputString.length()>0)
  {
     commandString = inputString.substring(1,5);
  }
}
```

Appendix 2: Quality Management System Program using Visual Studio 2015 (C#)

A. Initial Window C# Code

```
using…

namespace QMS_Application
{
    public partial class Form1 : Form
```

```
{
   public Form1()
   {
      InitializeComponent();
   }

   private void QMS_RUNbttn_Click(object sender, EventArgs e)
   {
      QMS_RUN_App runner = new QMS_RUN_App();  //Show the RUN Window

      runner.Show();

      Form1 main = new Form1();                  //Hide the initital Window
      main.Hide();
   }

   private void QMS_LEARNbttn_Click(object sender, EventArgs e)
   {
     QMS_LEARN_APP learner = new QMS_LEARN_APP();

     //Show the LEARN Window
     //Hide the initial window
  }

   private void EXITbttn_Click(object sender, EventArgs e)
```

```
        {
            //Close and Exit the application
        }

        private void button1_Click(object sender, EventArgs e)
        {
            ReportForm reporter = new ReportForm();

            //Show Report Window
            //Hide the initial window

        }
    }
}
```

B. Learn Mode C# Code

```
using…

namespace QMS_Application
{
    public partial class QMS_LEARN_APP : Form
    {
        public QMS_LEARN_APP()
        {
```

```
    InitializeComponent();
}

FilterInfoCollection filterinfoCollection;
VideoCaptureDevice captureDevice;

MySqlConnection conn = new MySqlConnection();
MySqlCommand cmd = new MySqlCommand();
MySqlDataAdapter da = new MySqlDataAdapter();
DataTable dt = new DataTable();

public string statusQR = "";
public int keepCount = 0;

private void QMS_LEARN_APP_Load(object sender, EventArgs e)
{
    this.Focus();

    conn.ConnectionString = "server=localhost;user
id=root;persistsecurityinfo=True;database=iqms_app;password=polaroid13";

    ResetGUI();
```

```
        filterinfoCollection = new
FilterInfoCollection(FilterCategory.VideoInputDevice);

        foreach (FilterInfo filterInfo in filterinfoCollection)
        {
            WebCamCMB.Items.Add(filterInfo.Name);
        }

        WebCamCMB.SelectedIndex = 0;
    }

    private void ScanBttn_Click(object sender, EventArgs e)
    {
        captureDevice = new
VideoCaptureDevice(filterinfoCollection[WebCamCMB.SelectedIndex].MonikerString);
        captureDevice.NewFrame += CaptureDevice_NewFrame;
        captureDevice.Start();
        timer1.Start();
    }

    private void CaptureDevice_NewFrame(object sender, NewFrameEventArgs
eventArgs)
    {
        ScanBox.Image = (Bitmap)eventArgs.Frame.Clone();
    }
```

```
    private void QMS_LEARN_APP_FormClosing(object sender,
FormClosingEventArgs e)
    {
        try
        {
            if (captureDevice.IsRunning)
                captureDevice.Stop();

            timer1.Stop();
        }
        catch
        { }

    }

    private void timer1_Tick(object sender, EventArgs e)
    {
        if (ScanBox.Image != null)
        {
            BarcodeReader Reader = new BarcodeReader();
            Result result = Reader.Decode((Bitmap)ScanBox.Image);

            if (result != null)
            {
                txtResult.Text = result.ToString();
```

```
            timer1.Stop();
            if (captureDevice.IsRunning)
                captureDevice.Stop();
            ScanBttn.Hide();
            bttnProcess.Show();
        }
    }
}

private void PASSrdb_CheckedChanged(object sender, EventArgs e)
{
    if (PASSrdb.Checked == true || FAILrdb.Checked == true)
        ScanBttn.Show();
    else
        ScanBttn.Hide();

    if (PASSrdb.Checked == true)
        statusQR = "PASS";

    keepCount = 0;

}

private void FAILrdb_CheckedChanged(object sender, EventArgs e)
{
```

```
        if (FAILrdb.Checked == true || PASSrdb.Checked == true)
            ScanBttn.Show();
        else
            ScanBttn.Hide();

        if (FAILrdb.Checked == true)
            statusQR = "FAIL";

        keepCount = 0;
    }

    private void bttnProceed_Click(object sender, EventArgs e)
    {
        // Save New Product Details: Product Type, Number of Stations, QR Code Separator, and Image Acquisition Station
        // To track New Products as Introduced to the System
        try
        {
            //create an insert query;
            string query = "INSERT INTO new_product_tracker (new_product_type,num_of_stations,qr_separator,acq_station_num, date_time) VALUES('" + txtProduct.Text + "','" + txtStations.Text + "','" + txtSeperator.Text + "','" + cmbImageCheck.Text + "','" + DateTime.Now + "')";

            //create table query
```

```
        string createT = "CREATE TABLE `iqms_app`.`" +
txtProduct.Text.ToLower() + "_learn` (`id_column1` VARCHAR(15) NOT NULL,
`newcolumn2` VARCHAR(15) NULL, `newcolumn3` VARCHAR(15) NULL,
`newcolumn4` VARCHAR(15) NULL, `newcolumn5` VARCHAR(15) NULL,
`qr_status` VARCHAR(15) NOT NULL, PRIMARY KEY (`id_column1`))";

        conn.Open();

        cmd.Connection = conn;
        cmd.CommandText = query;
        //execute the INSERT query
        int result = cmd.ExecuteNonQuery();

        //validate the result of the executed query
        if (result != 0)
        {
            MessageBox.Show("Data has been saved in the SQL database");

        }
        else
        {
            MessageBox.Show("SQL QUERY ERROR");
        }

        conn.Close();
```

```
    txtProduct.Enabled = false;
    txtStations.Enabled = false;
    cmbImageCheck.Enabled = false;
    txtSeperator.Enabled = false;

    label1.Show();
    PASSrdb.Show();
    FAILrdb.Show();

    //Create the table
    conn.Open();

    cmd.Connection = conn;
    cmd.CommandText = createT;
    //execute the CREATE TABLE query
    cmd.ExecuteNonQuery();
    conn.Close();
    bttnProceed.Enabled = false;

}
catch (Exception ex) //catch exeption
{

    //displaying error message.
```

```
            MessageBox.Show(ex.Message);
            conn.Close();
        }

    }

    private void bttnProcess_Click(object sender, EventArgs e)
    {
        // Save New Product Details from QR Code Captured
       try
        {
            String str = txtResult.Text;
            String[] strgs = str.Split(Convert.ToChar(txtSeperator.Text));

            string insertT = "INSERT INTO `iqms_app`.`" + txtProduct.Text.ToLower()
+ "_learn` (`id_column1`, `newcolumn2`, `newcolumn3`, `newcolumn4`,`newcolumn5`,
`qr_status`) VALUES ('" + strgs[0] + "','" + strgs[1] + "','" + strgs[2] + "','" + strgs[3] +
"','" + strgs[4] + "','" + statusQR + "')";

            conn.Open();
            cmd.Connection = conn;
            cmd.CommandText = insertT;

            int insertR = cmd.ExecuteNonQuery();

            //validate the result of the executed query
```

```
            if (insertR != 0)
            {
                MessageBox.Show("Data has been saved in the SQL database");
            }
            else
            {
                MessageBox.Show("SQL QUERY ERROR");
            }

            conn.Close();
            ScanBttn.Show();
            txtResult.Clear();
            bttnProcess.Hide();
            keepCount++;

            if (keepCount == 3)
            {
                FAILrdb.Checked = false;
                PASSrdb.Checked = false;
               // bttnProcess.Enabled = false;

                MessageBox.Show("The Minimum of 3 QR Codes Scanned. Select
Pass/Fail to Continue OR Select Exit/Run.");
            }
        }
```

```
        catch (Exception ex)
        {
            //displaying error message
            MessageBox.Show(ex.Message);
            conn.Close();
        }

    }

    public Random a = new Random(); // replace from new
Random(DateTime.Now.Ticks.GetHashCode());
                                    // Since similar code is done in default constructor internally
    public List<int> randomList = new List<int>();
    int MyNumber = 0;

    private void NewNumber()
    {
        MyNumber = a.Next(0, 100);
        while (randomList.Contains(MyNumber))
            MyNumber = a.Next(0, 100);
    }

    private void txtNewProduct_Click(object sender, EventArgs e)
    {
        ResetGUI();
```

```
    }

    private void ResetGUI()
    {
        txtProduct.Enabled = true;
        txtProduct.Clear();
        txtStations.Enabled = true;
        txtStations.Clear();
        cmbImageCheck.Enabled = true;
        cmbImageCheck.SelectedItem = null;
        cmbImageCheck.ResetText();
        txtSeperator.Enabled = true;
        txtSeperator.Clear();

        bttnProceed.Enabled = false;

        ScanBttn.Hide();
        bttnProcess.Hide();

        label1.Hide();
        PASSrdb.Hide();
        FAILrdb.Hide();
    }

}
```

}

C. Run Mode C# Code

```
using…

namespace QMS_Application
{
   public partial class QMS_RUN_App : Form
   {

      public QMS_RUN_App()
      {
         InitializeComponent();

      }

      //bool isConnected = false;
      //String[] ports;
      //SerialPort port;

      delegate void SetTextCallback(string text);

      MySqlConnection conn = new MySqlConnection();
      MySqlCommand cmd = new MySqlCommand();
```

```
MySqlDataAdapter da = new MySqlDataAdapter();
DataTable dt = new DataTable();


public string lastProduct = "";
public string trackRunS = "default";
string sql;
int IA_point = 0;


string ardACKN = "#ACKN\n";
string ardDATAreq = "#DATA\n";


string GenID = "";
bool FirstID = true;
string Data_s1 = "";
string Data_s2 = "";
string Data_s3 = "";
string dataSub = "";
string data_quality = "";


string COMPortOpen = "";


private void QMS_RUN_App_Load(object sender, EventArgs e)
{
    this.Focus();
```

```
        conn.ConnectionString = "server=localhost;user
id=root;persistsecurityinfo=True;database=iqms_app;password=polaroid13";

        //disableControls();


        getAvailableCOMPorts();

        loadDefault();

    }

    void getAvailableCOMPorts()
    {
        lbSerialPort.Items.Clear();

        foreach (string item in System.IO.Ports.SerialPort.GetPortNames())
        {
            //store each retrieved available port name in the Listbox...
            lbSerialPort.Items.Add(item);
            lbSerialPort.Sorted = true;

            //Remove Bluetooth Serial Ports
            lbSerialPort.Items.Remove("COM14");
            lbSerialPort.Items.Remove("COM15");
```

```
    }
}

public string ExecuteQuery(string query, MySqlCommand command)
{
    command.CommandType = CommandType.Text;
    command.CommandText = query;
    object i = cmd.ExecuteScalar();
    return i.ToString();
}

private void button1_Click(object sender, EventArgs e)
{
    if (trackRunS == "Loaded")
    {

        loadDefault();

        bttnRunSession.Text = "LOAD ALTERNATIVE PRODUCT";
        trackRunS = "default";

    }
    else if (trackRunS == "default")
    {
```

```
        sql = "SELECT new_product_type FROM new_product_tracker";


        LoadCombo(sql);


        bttnRunSession.Text = "RUN LAST LEARNED ASSEMBLY";
        trackRunS = "Loaded";
    }
}


private void LoadCombo(string sql)
{
    try
    {
        cmbBox.Items.Clear();
        cmbBox.ResetText();
        cmbBox.Enabled = true;
        conn.Open();
        MySqlCommand cmd = new MySqlCommand(sql, conn);
        MySqlDataReader reader = cmd.ExecuteReader();
        while (reader.Read())
        {
            string name = reader.GetString("new_product_type");
            cmbBox.Items.Add(name);
        }
```

```
            cmd.Dispose();
            reader.Close();

            lbSerialPort.Items.Clear();
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);

        }
        finally
        {
            conn.Close();

        }
    }

    public void loadDefault()
    {
        //Retrieve the last entry into the product tracker table ProductType
        try
        {
            cmbBox.Items.Clear();
            cmbBox.ResetText();
            cmbBox.Enabled = false;
```

```
            string query = "SELECT new_product_type FROM new_product_tracker
WHERE date_time=(SELECT max(date_time) FROM new_product_tracker)";
            string getProductT;

            conn.Open();

            cmd.Connection = conn;
            cmd.CommandText = query;

            getProductT = ExecuteQuery(query, cmd);

            txtProductType.Text = getProductT;

            getAvailableCOMPorts();

            bttnRun.Enabled = true;

        }
        catch (Exception exc)
        {
            MessageBox.Show(exc.Message);
        }
        finally
        {
```

```
            conn.Close();


        }


        getIAvalue();
    }


    private void getIAvalue()
    {
        try
        {
            string query = "SELECT acq_station_num FROM new_product_tracker
WHERE date_time=(SELECT max(date_time) FROM new_product_tracker)";
            string IAstring;
            conn.Open();


            cmd.Connection = conn;
            cmd.CommandText = query;


            IAstring = ExecuteQuery(query, cmd);
            IAstring.Trim();


            IA_point = int.Parse(IAstring);
        }
        catch(Exception exs)
```

```
    {
        MessageBox.Show(exs.Message);
    }
    finally
    {
        conn.Close();
    }

}

private void bttnRun_Click(object sender, EventArgs e)
{

    if (!serialPort1.IsOpen)
    {
        serialPort1.Open(); //Open COM3 and wait for  Hardware Node
    }
    bttnStopiQMS.Enabled = true;

    bttnRun.Enabled = false;

}

private void cmbBox_SelectedIndexChanged(object sender, EventArgs e)
{
```

```
        txtProductType.Text = cmbBox.SelectedItem.ToString();
    }

    private void bttnStopiQMS_Click(object sender, EventArgs e)
    {
        if (serialPort1.IsOpen) { serialPort1.Close(); }

        if (serialPort2.IsOpen) { serialPort2.Close(); }

        if (serialPort3.IsOpen) { serialPort3.Close(); }

        if(conn.State == ConnectionState.Open)
        {
            conn.Close();
        }

        loadDefault();

    }

//HANDLE SERIAL PORT DATA RECEIVED CODE HERE

    private void SetText(string text)
    {
        // InvokeRequired compares the thread ID of the
```

```
    // calling thread to the thread ID of the creating thread.
    // If these threads are different, it returns true.
    if (this.txtProductType.InvokeRequired)
    {
        SetTextCallback d = new SetTextCallback(SetText);
        this.Invoke(d, new object[] { text });
    }
    else
    {
        this.txtCOM.Text = text;

        string newString = GetStringBetweenCharacters(text, '#', '@');

        txtCOM.Text = newString;

    }
}

private void sendACKN2ARDUINO(string ardText)
{
    if(ardText == "RDY" && COMPortOpen == "COM3")
    {
        serialPort1.Write(ardACKN);
    }
    else if(ardText == "RDY" && COMPortOpen == "COM4")
```

```
    {
        serialPort2.Write(ardACKN);
    }
    else if(ardText == "RDY" && COMPortOpen == "COM5")
    {
        serialPort3.Write(ardACKN);
    }
}

private void sendRequest(string stbyText)
{
    if(stbyText == "STBY" && COMPortOpen == "COM3")
    {
        serialPort1.Write(ardDATAreq);
    }
    else if (stbyText == "STBY" && COMPortOpen == "COM4")
    {
        serialPort2.Write(ardDATAreq);
    }
    else if (stbyText == "STBY" && COMPortOpen == "COM5")
    {
        serialPort3.Write(ardDATAreq);
    }
}
```

```
public static string GetStringBetweenCharacters(string input, char charFrom, char charTo)
{
    int posFrom = input.IndexOf(charFrom);
    if (posFrom != -1) //if found char
    {
        int posTo = input.IndexOf(charTo, posFrom + 1);
        if (posTo != -1) //if found char
        {
            return input.Substring(posFrom + 1, posTo - posFrom - 1);
        }
    }

    return string.Empty;
}

private void txtCOM_TextChanged(object sender, EventArgs e)
{
    string ardReply = txtCOM.Text;

    if (txtCOM.Text == "RDY")
    {
        sendACKN2ARDUINO(ardReply);
    }
    else if (txtCOM.Text == "STBY" && GenID != "")
```

```
    {
        sendRequest(ardReply);
    }
    else if(txtCOM.Text == "STBY" && GenID == "")
    {
        generateIDunique();
        sendRequest(ardReply);
    }
    else if(txtCOM.Text.Contains("Digital:") || ardReply.Contains("Analog:"))
    {
        processData(ardReply);
    }

}

private void generateIDunique()
{
    string newIDnum = "";
    string newIDalph = "";
    int num = 0;

    try
    {

        string query = "";
```

```
string getIDCol;


if (FirstID == true)
{
    query = "SELECT id_column1 FROM " + txtProductType.Text.ToLower()
+ "_learn WHERE id_column1 =(SELECT max(id_column1) FROM " +
txtProductType.Text.ToLower() + "_learn);";
}
else if (FirstID == false)
{
    query = "SELECT prod_ID FROM " + txtProductType.Text.ToLower() +
"_run WHERE prod_ID =(SELECT max(prod_ID) FROM " +
txtProductType.Text.ToLower() + "_run);";
}



conn.Open();


cmd.Connection = conn;
cmd.CommandText = query;


getIDCol = ExecuteQuery(query, cmd);


newIDnum = Regex.Replace(getIDCol, "[^0-9.]", "");
newIDalph = Regex.Replace(getIDCol, @"[\d-]", ""); ;
```

```
            num = int.Parse(newIDnum);


            num++;


            GenID = newIDalph.ToString() + num.ToString();
            FirstID = false;


        }
        catch(Exception excc)
        {
            MessageBox.Show(excc.Message);
        }
        finally
        {
            conn.Close();
        }



}

private void processData(string ArdReply)
{
    ArdReply = txtCOM.Text;

    if (ArdReply != "")
```

```
{
    dataSub = GetStringBetweenCharacters(ArdReply, '#', '@');

    ArdReply = dataSub;
}

try
{
    //Strip down data; Remove all except the Digital or Analog data
        if(ArdReply.Contains("Digital: "))
    {
        dataSub = ArdReply.Replace("Digital: ", "");
        dataSub.Trim();
    }
    else if(ArdReply.Contains("Analog: "))
    {
        dataSub = ArdReply.Replace("Analog: ", "");
        dataSub.Trim();
    }

    //Assign Data to Station (1 to 3)
    if (serialPort1.IsOpen && COMPortOpen == "COM3")
    {
        if (Data_s1 == "")
        { Data_s1 = dataSub; }
```

```
            if (serialPort1.IsOpen)
            {
                Thread CloseDown = new Thread(new ThreadStart(CloseSerialIfOpen));
//close port in new thread to avoid hang
            }

            Thread.Sleep(1000);

            if (!serialPort2.IsOpen)
            { serialPort2.Open(); }
            // txtCOM.Clear();
        }
        else if (serialPort2.IsOpen && COMPortOpen == "COM4")
        {
            if (Data_s2 == "")
            { Data_s2 = dataSub; }

            Thread.Sleep(1000);
            if (serialPort2.IsOpen)
            {
                Thread CloseDown = new Thread(new ThreadStart(CloseSerialIfOpen));
//close port in new thread to avoid hang
            }
```

```
            Thread.Sleep(1000);
            if (!serialPort3.IsOpen)
            { serialPort3.Open(); }
           // txtCOM.Clear();
        }
        else if (serialPort3.IsOpen && COMPortOpen == "COM5")
        {
            if (Data_s3 == "")
            { Data_s3 = dataSub; }

            Thread.Sleep(1000);
            if (serialPort3.IsOpen)
            {
                Thread CloseDown = new Thread(new ThreadStart(CloseSerialIfOpen));
//close port in new thread to avoid hang
            }

            Thread.Sleep(1000);
            if (!serialPort1.IsOpen)
            { serialPort1.Open(); }
            //txtCOM.Clear();
        }

        //Save all data once acquired
        if(Data_s1 != "" && Data_s2 != "" && Data_s3 != "")
```

```
        {
            DataCollected();
        }


    }
    catch(Exception exc)
    {
        MessageBox.Show(exc.Message);
    }
}

private void CloseSerialIfOpen()

{

    try

    {

        if (serialPort1.IsOpen) { serialPort1.Close(); }//close the serial port

        if (serialPort2.IsOpen) { serialPort2.Close(); }

        if (serialPort3.IsOpen) { serialPort3.Close(); }
```

```
    }

    catch (Exception ex)

    {

        MessageBox.Show(ex.Message); //catch any serial port closing error messages

    }

}



// private void NowClose(object sender, EventArgs e)

// {

//    this.Close(); //now close the form

// }

private void DataCollected()

{

    //Compare data and verify FAIL or PASS
```

```
        createTableifNone();

        saveDATAcollected();

        assignQualityVariable();

    }

    private void createTableifNone()
    {
        try
        {
            string createT = "CREATE TABLE IF NOT EXISTS `iqms_app`.`" +
txtProductType.Text.ToLower() + "_run` (`prod_id` VARCHAR(15) NOT NULL,
`product_type` VARCHAR(15) NULL, `newcolumn3` VARCHAR(15) NULL,
`newcolumn4` VARCHAR(15) NULL, `newcolumn5` VARCHAR(15) NULL,
`qr_status` VARCHAR(15) NULL, `product_date` VARCHAR(45) , PRIMARY KEY
(`prod_id`))";

            conn.Open();

            cmd.Connection = conn;
            cmd.CommandText = createT;
            //execute the INSERT query
            int result = cmd.ExecuteNonQuery();
```

```
            //validate the result of the executed query
            if (result != 0)
            {
                MessageBox.Show("Success: Table Created");
            }



        }
        catch(Exception excp)
        {
            MessageBox.Show(excp.Message);
        }
        finally
        {
            conn.Close();
        }
    }

    private void saveDATAcollected()
    {
        try
        {
            string insertT = "INSERT INTO `iqms_app`.`" +
txtProductType.Text.ToLower() + "_run` (`prod_id`, `product_type`, `newcolumn3`,
`newcolumn4`, `newcolumn5`, `product_date`) VALUES ('" + GenID.ToString() + "','"
+ txtProductType.Text + "','" + Data_s1.ToString() + "','" + Data_s2.ToString() + "','"
```

```
+ Data_s3.ToString() + "'," + DateTime.Now.ToString() + "') ON DUPLICATE KEY
UPDATE `product_type` = `product_type`";

            conn.Open();

            cmd.Connection = conn;
            cmd.CommandText = insertT;

            int insertR = cmd.ExecuteNonQuery();

            //validate the result of the executed query
            if (insertR != 0)
            {
                MessageBox.Show("Product Data has been saved in the SQL database");
            }
            else
            {
                MessageBox.Show("SQL QUERY ERROR");
            }

            conn.Close();

        }
        catch(Exception exc1)
        {
```

```
        MessageBox.Show(exc1.Message);
    }

}

private void assignQualityVariable()
{
    bool c1_state = false;
    bool c2_state = false;
    bool c3_state = false;

    string[] min_value = new string[3];

    string[] max_value = new string[3];

    string updateR = "";

    try
    {
        //Query Table for PASS

        min_value = PassVariablesLower();

        max_value = PassVariablesUpper();
```

```
//Workout/Calculate PASS Determinant (Pd)

c1_state = DeterminePass(min_value[0], max_value[0], Data_s1);

c2_state = DeterminePass(min_value[1], max_value[1], Data_s2);

c3_state = DeterminePass(min_value[2], max_value[2], Data_s3);

//Assign P/F Quality Variable
if (c1_state == true && c2_state == true && c3_state == true)
{ data_quality = "PASS"; }
else
{ data_quality = "FAIL"; }

updateR = "UPDATE `iqms_app`.`" + txtProductType.Text + "_run` SET
`qr_status` = '" + data_quality.ToString() + "' WHERE (`prod_id` = '" +
GenID.ToString() + "');";

conn.Open();

cmd.Connection = conn;
cmd.CommandText = updateR;

int qUpdate = cmd.ExecuteNonQuery();
```

```
        //validate the result of the executed query
        if (qUpdate != 0)
        {

            MessageBox.Show("Product Data Q-Status has been updated");

        }
        else
        {
            MessageBox.Show("SQL QUERY ERROR");
        }

    }
    catch(Exception exc)
    {
        MessageBox.Show(exc.Message);

    }
}

private string[] PassVariablesLower()
{
    string[] min_values = new string[3];

    try
```

```
{
    //first column query
    string minquery = "SELECT DISTINCT newcolumn3 FROM " +
txtProductType.Text.ToLower() + "_learn WHERE newcolumn3 =(SELECT
min(newcolumn3) FROM " + txtProductType.Text.ToLower() + "_learn WHERE
qr_status = 'PASS');";

    conn.Open();

    cmd.Connection = conn;
    cmd.CommandText = minquery;

    min_values[0] = ExecuteQuery(minquery, cmd);

    conn.Close();

   //2nd column query
    string minquery2 = "SELECT DISTINCT newcolumn4 FROM " +
txtProductType.Text.ToLower() + "_learn WHERE newcolumn4 =(SELECT
min(newcolumn4) FROM " + txtProductType.Text.ToLower() + "_learn WHERE
qr_status = 'PASS');";

    conn.Open();

    cmd.Connection = conn;
    cmd.CommandText = minquery2;
```

```
            min_values[1] = ExecuteQuery(minquery2, cmd);


            conn.Close();


            //3rd column query
            string minquery3 = "SELECT DISTINCT newcolumn5 FROM " +
txtProductType.Text.ToLower() + "_learn WHERE newcolumn5 =(SELECT
min(newcolumn5) FROM " + txtProductType.Text.ToLower() + "_learn WHERE
qr_status = 'PASS');";


            conn.Open();


            cmd.Connection = conn;
            cmd.CommandText = minquery3;


            min_values[2] = ExecuteQuery(minquery3, cmd);


            conn.Close();


        }
        catch (Exception exx)
        {
            MessageBox.Show(exx.Message);
        }
        finally
```

```
        {
            if (conn.State == ConnectionState.Open) { conn.Close(); }
        }

            return min_values;

    }

    private string[] PassVariablesUpper()
    {
        string[] max_values = new string[3];

        try
        {
            //first column query

            string maxquery = "SELECT DISTINCT newcolumn3 FROM " +
txtProductType.Text.ToLower() + "_learn WHERE newcolumn3 =(SELECT
max(newcolumn3) FROM " + txtProductType.Text.ToLower() + "_learn WHERE
qr_status = 'PASS');";

            conn.Open();

            cmd.Connection = conn;
            cmd.CommandText = maxquery;
```

max_values[0] = ExecuteQuery(maxquery, cmd);

conn.Close();

//2nd column query

string maxquery2 = "SELECT DISTINCT newcolumn4 FROM " + txtProductType.Text.ToLower() + "_learn WHERE newcolumn4 =(SELECT max(newcolumn4) FROM " + txtProductType.Text.ToLower() + "_learn WHERE qr_status = 'PASS');";

conn.Open();

cmd.Connection = conn;

cmd.CommandText = maxquery2;

max_values[1] = ExecuteQuery(maxquery2, cmd);

conn.Close();

//3rd column query

string maxquery3 = "SELECT DISTINCT newcolumn5 FROM " + txtProductType.Text.ToLower() + "_learn WHERE newcolumn5 =(SELECT max(newcolumn5) FROM " + txtProductType.Text.ToLower() + "_learn WHERE qr_status = 'PASS');";

conn.Open();

```
        cmd.Connection = conn;
        cmd.CommandText = maxquery3;

        max_values[2] = ExecuteQuery(maxquery3, cmd);

        conn.Close();

    }
    catch (Exception exx)
    {
        MessageBox.Show(exx.Message);
    }
    finally
    {
        if (conn.State == ConnectionState.Open) { conn.Close(); }
    }

    return max_values;

}

private bool DeterminePass(string min_value, string max_value, string d_S)
{
    bool result = false;
```

```
int dSub = 0;
string minSub = "";
int minSubnum = 0;

string maxSub = "";
int maxSubnum = 0;



try
{

    dSub = Convert.ToInt32(d_S.Substring(0, d_S.IndexOf('.') > 0 ?
d_S.IndexOf('.') : d_S.Length));

    minSub = Regex.Replace(min_value, "[^0-9.]", "");
    minSubnum = Int32.Parse(minSub, NumberStyles.AllowThousands);

    maxSub = Regex.Replace(max_value, "[^0-9.]", "");
    maxSubnum = Int32.Parse(maxSub, NumberStyles.AllowThousands);

    if (dSub >= minSubnum && dSub <= maxSubnum)
    { result = true; }
}
catch(Exception excc)
{
```

```
            MessageBox.Show(excc.Message);
        }
        return result;
    }
  }
}
```