

5-2014

DESIGN OF A CUSTOM SOFTWARE APPLICATION TO MONITOR AND COMMUNICATE CNC MACHINING PROCESS INFORMATION TO AID IN CHATTER IDENTIFICATION

Valerie Pezzullo

Clemson University, valerie.pezzullo@gmail.com

Follow this and additional works at: https://tigerprints.clemson.edu/all_theses



Part of the [Computer Sciences Commons](#), [Mechanical Engineering Commons](#), and the [Operations Research, Systems Engineering and Industrial Engineering Commons](#)

Recommended Citation

Pezzullo, Valerie, "DESIGN OF A CUSTOM SOFTWARE APPLICATION TO MONITOR AND COMMUNICATE CNC MACHINING PROCESS INFORMATION TO AID IN CHATTER IDENTIFICATION" (2014). *All Theses*. 1932.

https://tigerprints.clemson.edu/all_theses/1932

This Thesis is brought to you for free and open access by the Theses at TigerPrints. It has been accepted for inclusion in All Theses by an authorized administrator of TigerPrints. For more information, please contact kokeefe@clemson.edu.

DESIGN OF A CUSTOM SOFTWARE APPLICATION TO MONITOR AND
COMMUNICATE CNC MACHINING PROCESS INFORMATION
TO AID IN CHATTER IDENTIFICATION

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
Mechanical Engineering

by
Valerie Nicole Pezzullo
May 2014

Accepted by:
Dr. Laine Mears, Committee Co-Chair
Dr. Georges Fadel, Committee Co-Chair
Dr. Harry Law
Dr. John Wagner

ABSTRACT

In any manufacturing environment, it is important to be able to monitor the Computer Numerical Control (CNC) machining process so that high quality parts can be produced in the least amount of time in order to be profitable. This involves acquiring the proper parameters needed from the machine's controller, which can prove to be difficult with proprietary machine tools that tend to limit access to the internal data collected by the controller. This closed approach to controller design also means that many technological advances that have recently become prevalent in society are not being adopted in the manufacturing industry, preventing the interoperability between hardware and software components and adding to the shortcomings in communicating the necessary machining parameters to machine operators. The project described in this thesis offers a solution to some of the communication, productivity, and part quality problems in the American manufacturing industry by providing a custom software application that integrates MTConnect, an emerging interoperable data communication standard, with proprietary data acquisition tools and custom sensors to monitor and communicate CNC machining process information.

The application described in this thesis was designed to aid in the identification of chatter conditions to the machine operator and to other users to take action for chatter suppression and avoidance. Chatter is an undesirable phenomenon that can reduce part quality and increase tool wear. These consequences result in higher costs to replace damaged parts and tools as well as increasing the amount of machine downtime which can reduce a company's overall productivity. Once chatter is detected in the audible

frequency range, damage to the workpiece has already occurred. Therefore, an early identification and communication method with the machine tool is warranted to easily monitor the machine in the event of impending dynamic part damage. This application was developed to provide a means to monitor cutting conditions to reduce and prevent chatter in the machining process and to aid in analysis to avoid subsequent unstable operating conditions.

Preserving part quality and productivity in manufacturing is also dependent on accurate information provided about the specific parts involved in the machining process. In addition to monitoring the process, this application facilitates the communication of part-specific information by improving the input and tracking of part numbers, and organizes the machining process information in a central location according to the specific part. Improving the part tracking process can aid in the organization of data to analyze the machining process for increased quality in future operations.

The application can also be customized for other implementations, which can benefit many different industrial manufacturing facilities as well as academics in performing experimental research. It is important for the manufacturing industry and its partners in academia to be able to bridge the communication gap to increase the knowledge of the machining process and therefore manufacturing productivity and profitability.

DEDICATION

I would like to dedicate this thesis to my family and friends who have continuously supported me throughout my life. A special dedication to my parents Vincent and Suzanne Pezzullo who have always believed in me and pushed me to do what I thought was impossible. Without their support and encouragement through both the good and bad times, I would not be where I am today. Another special dedication to my fiancé James Fadool for his love and support throughout my graduate career and for continuing to challenge me to think outside the box.

ACKNOWLEDGMENTS

I would like to thank my research advisor and committee co-chair Dr. Laine Mears for his guidance and support throughout my graduate studies and for providing me with opportunities that have helped me start my engineering career successfully. I would also like to thank Dr. Georges Fadel for agreeing to serve as my committee co-chair along with Dr. Mears, and I would like to thank Dr. Harry Law and Dr. John Wagner for serving on my committee. I greatly appreciate all of the feedback that the committee provided on my work throughout my studies. I would also like to thank the Department of Automotive Engineering staff including Frank Webb, David Mann, and Gary Mathis for their assistance with laboratory equipment. Finally I would like to thank the Department of Mechanical Engineering faculty, especially Dr. Todd Schweisinger, for providing me with a teaching assistantship so that I could attend graduate school and further my education.

TABLE OF CONTENTS

	Page
TITLE PAGE	i
ABSTRACT	ii
DEDICATION	iv
ACKNOWLEDGMENTS	v
LIST OF FIGURES	viii
CHAPTER	
I. INTRODUCTION	1
Motivation	1
Research Focus Areas	5
Problem Statement	6
Objectives	7
Organization	8
II. BACKGROUND	9
History of Communication in Manufacturing	10
System Interoperability	13
Introduction to MTConnect	16
MTConnect Communication Process	18
Previous Work with MTConnect	25
MTConnect Challenge	28
Part Quality Preservation	30
III. MACHINE, HARDWARE, AND SOFTWARE SPECIFICATIONS	34
CNC Machine	34
Custom Integrated Tooling Sensors	36
Data Acquisition Hardware	37
Data Acquisition Software	41
MTConnect Installation and Setup	47

Table of Contents (Continued)

	Page
IV. SOFTWARE APPLICATION DEVELOPMENT AND ARCHITECTURE	51
Software Development Tools	51
Application Functions and Architectures	57
V. SYSTEM OPERATIONAL PROCEDURE FOR THE CHATTER IDENTIFICATION APPLICATION	74
System Data Communication Process	74
Operational Procedure	76
User-Interface Operation	80
VI. APPLICATION IMPLEMENTATIONS FOR CHATTER IDENTIFICATION	92
Chatter Identification Techniques	92
Chatter Simulation	93
Spectral Analysis in LabVIEW	94
Results in Chatter Identification Application	101
VII. RECOMMENDATIONS FOR FUTURE WORK	107
Limitations and Shortcomings	107
Long-Term Implementation Vision	114
Benefits to Manufacturing Industry	116
VIII. CONCLUSION.....	126
REFERENCES	129

LIST OF FIGURES

Figure		Page
1.1	A breakdown of the focus areas of the project.	5
2.1	The basic architecture of an ideal MTConnect system.	19
2.2	The MTConnect schema when the “Probe” command is sent to the sample agent at “http://agent.mtconnect.org”. The red box shows an example of a component, the rotary “C” axis, and one of its associated DataItems, which is defined as the “spindle speed” with the units of “revolution/minute”.	22
2.3	The MTConnect schema when the “Current” command is sent to the sample agent at “http://agent.mtconnect.org”. The green box shows a “ComponentStream”, which is a stream of data from a component, of the rotary “C” axis and the corresponding values of a “Sample” defined as the “Spindle Speed” DataItem. The red box highlights the current value of the actual spindle speed.	23
2.4	The MTConnect schema when the “Sample” command is sent to the sample agent at “http://agent.mtconnect.org”. The green box shows the “ComponentStream” of the “Path” component, and the red box shows the stream of samples of the “Frt”, or feed-rate, DataItem with associated values along with a timestamp to show the time at which the agent was sampled to retrieve this data.	24
2.5	The user interface of the “Probe Direction Analyzer” application.	26
2.6	The user interface of the “PneuViz” application.	27
2.7	The user interface of the “Quality Monitoring” application.	27
2.8	A schematic of the “wavy profile” generated by a vibrating tool.	31

List of Figures (Continued)

Figure	Page
2.9 In-phase wavy profiles generates a negligible chip thickness from one revolution to the next (left) and out-of-phase wavy profiles generate a variable chip thickness and therefore variable cutting force from one revolution to the next (right).	32
2.10 A simplified representation of a typical stability lobe diagram.	33
3.1 Okuma SPACE TURN LB4000EX 2-axis horizontal CNC lathe with THINC OSP-P200L controller.	35
3.2 NI CompactRIO chassis with reconfigurable FPGA and Real-Time capabilities and removable I/O modules.	38
3.3 NI cRIO-9022 chassis with several different I/O modules in place.	39
3.4 NI 9215 Analog Input module with either BNC connection terminals (left) or screw terminal connections (right).	40
3.5 NI 9211 Thermocouple Input module.	40
3.6 Screenshot of the LabVIEW project explorer.	43
3.7 Screenshot of the Front Panel (left) and Block Diagram (right) components of the VI on the FPGA level.	44
3.8 Screenshots of the Front Panel (left) and Block Diagram (right) components of the VI on the Real-Time level.	44
3.9 NI Distributed System Manager shows the values of each shared variable.	45
3.10 Screenshot of the Front Panel (left) and Block Diagram (right) components of the VI on the host PC level.	45
3.11 Screenshot of the NI-MAX tool which shows the connections to various NI hardware.	46
3.12 Screenshot of MTConnect adapter GUI on the machine PC.	48

List of Figures (Continued)

Figure	Page
4.1 Several IDE options considered to develop the application based on certain criteria that were necessary for its proper functionality.	52
4.2 Screenshot of the Visual Studio 2008 IDE showing the C# code development tab (center) and Solution Explorer (top right).	55
4.3 Screenshot of the Visual Studio 2008 IDE showing the “Windows form” design (center), Properties box (bottom right), and Toolbox (left).	56
4.4 Intermec SR61T barcode scanner for the GE Work Setup application.	58
4.5 Architecture and logic flow of the Work Setup application requested by GE Power & Water.	59
4.6 Barcode scanner user-interface for the GE Work Setup application.	60
4.7 Pop-up message to prompt the user to scan their badge.	61
4.8 “Scan Info” tab shows the database table that stores the badge number, job number, and part number after each scan.	62
4.9 “Part Info” tab shows the database table that stores the part file and work instruction file locations based on each part number. The application reads this table to search for a matching part number based on the part barcode that was scanned and opens the corresponding part file and work instruction file.	62
4.10 “Scan Info” table in the MS Access database.	63
4.11 “Part Info” table in the MS Access database.	63
4.12 Pop-up message to prompt the user to scan the job barcode.	64
4.13 Pop-up message to prompt the user to scan the part barcode.	65
4.14 Architecture and logic flow of MTConnect Sample Client.	66

List of Figures (Continued)

Figure	Page
4.15 “Agent Address” text box and “Connect” button. The user typed “agent.mtconnect.org” into the text box and would then click “Connect” to retrieve data from the agent.	67
4.16 The Agent Structure is displayed in a tree view (top) and the “c2 (Sspeed)” data item’s associated values and properties are displayed in the list view (bottom).	68
4.17 Data items can be dragged and dropped to this list view as well which will gather the information desired to be saved.	68
4.18 The plot displays each data item that is dragged and dropped into it as a separate series.	69
4.19 Screenshot of the application’s user-interface.	70
4.20 Overall application architecture separated into its major functions and sub-functions	71
5.1 Schematic of the data communication flow between various components of the system.	75
5.2 NI-MAX showing the connection to the cRIO chassis named “XYTABLE”.	78
5.3 A screenshot of the background LabVIEW Real-Time VI that is required for the external sensor data acquisition.	79
5.4 Screenshot of application running during machining operation.	80
5.5 Schematic of the application’s operational procedures.	81
5.6 After the “Start New Job” button is pressed, the job number is increased and shown in the text box (this would be Job Number 1). A message box pops up to prompt the user to scan the part barcode.	82

List of Figures (Continued)

Figure	Page
5.7	The job number and part number, which is entered after the part barcode is scanned, for the current job. 82
5.8	The “Save Data” section contains the automatically created file name, user chosen file extension (shown here as “.csv”) and the chosen file path. The user would click the “Browse...” button to choose the file location. When ready, the user can check the “Record data to file” box to save the data, and uncheck to stop saving data. 83
5.9	This message box pops up to notify the user of the chosen file path location. 84
5.10	The agent address of the Okuma LB4000EX machine tool is entered into the text box and the “Connect” button is clicked to connect to the agent. The “Disconnect” button can be pressed to disconnect from the agent and clear all lists and plots. 84
5.11	This tree view (top) shows the structure of the MTConnect agent that retrieves data from the adapter on the Okuma LB4000EX machine tool. The list view (below) shows a specific data item’s properties and value. This specific data item illustrated is the spindle speed with a value of 300 and units of “revolution/minute”. 85
5.12	MTConnect data items can be dragged into the “Machining Process Live Data” plot (top left) and the list view (bottom left) as well as the “Spindle Speed” gauge (right), which is a Measurement Studio indicator, to display the appropriate values. 86
5.13	The “Force Sensor Live Data” waveform graph and the “Tool Temperature” thermometer are Measurement Studio indicators that accept data from the shared variables originating from the background LabVIEW program connecting to the CompactRIO. 89

List of Figures (Continued)

Figure	Page
5.14 A screenshot of the .csv file in Microsoft Excel with data from the machining process collected through the application. The user can save any combination of data items along with data collected from each sensor and can plot this data for analysis. The filename reflects the job number and specific part number so the user can keep track of the part-specific machining process data.	90
6.1 Vibration shaker table.	93
6.2 Amplitude and Phase Spectrum PtbyPt sub-VI used in the spectral analysis of the force sensor.	96
6.3 The single-sided amplitude spectrum plot with a sample length of 200 (top left: normal conditions, top right: chatter conditions) from the unfiltered force sensor signal (bottom). Chatter was simulated on the force sensor as seen by the large variations in force at approximately 0.7 seconds, after which the amplitude spectrum sub-VI updated and displayed the frequency domain signal on the plot.	97
6.4 The single-sided amplitude spectrum plot with a sample length of 1000 (top left: normal conditions, top right: chatter conditions) from the unfiltered force sensor signal (bottom). Chatter was simulated on the force sensor as seen by the large variations in force at approximately 0.5 seconds, after which the amplitude spectrum sub-VI updated and displayed the frequency domain signal on the plot.	98
6.5 The single-sided amplitude spectrum plot with a sample length of 500 (top left: normal conditions, top right: chatter conditions) from the unfiltered force sensor signal (bottom). Chatter was simulated on the force sensor as seen by the large variations in force at approximately 0.8 seconds, after which the amplitude spectrum sub-VI updated and displayed the frequency domain signal on the plot.	99

List of Figures (Continued)

Figure	Page
6.6 The chatter identification application connected to the MTConnect agent on the Okuma machine simulator and the CompactRIO to acquire the force sensor data. The top figure shows the lateral force signal in the time domain, where the bottom shows the lateral force signal in the frequency domain.	102
6.7 The chatter identification application connected to the MTConnect agent on the Okuma machine simulator and the CompactRIO to acquire the force sensor data. The top figure shows the vertical force signal in the time domain, where the bottom shows the vertical force signal in the frequency domain.	103
6.8 The screenshot of the .csv file opened in Excel shows the force variations in the top plot and the Z-axis position of the tool in the bottom plot. The user can identify where chatter occurred as the tool moved in the Z-direction on the part. The user can also look at other data items that were saved to identify what parameters could have caused the chatter condition.	105

CHAPTER ONE

INTRODUCTION

This chapter introduces the motivation for the project, the areas in which this project is focused, the problem statement that was defined based on the needs of the manufacturing industry, and the project objectives that would serve as a requirements list to provide a solution to the problem statement and therefore a solution to the needs of the manufacturing industry.

Motivation

In recent years, there has been a decline in manufacturing of products in America [48]. Part of the reason for this is due to the downfalls in data communication in the industry. There has been a tremendous advancement in technology in the past few decades, from the original computers that required rooms full of equipment to the advanced processing and communication capabilities of current smartphones that put powerful computing power in the hands of individuals. The manufacturing industry has been slow to implement these advances in technology and is struggling with the ability to communicate important data from the manufacturing processes being performed.

The current technology available in our society allows for complex data streaming with devices such as smartphones, tablets, and products such as Google Chromecast that allows users to stream videos and applications from their PC, tablet, or smartphone to their television. These devices communicate data through standard interfaces and protocols such as USB, HTTP, TCP/IP, and other technologies that are widely known and

commonly used. These technologies would be extremely useful in the manufacturing industry to communicate information, but there is a lack of development of equipment that utilizes these types of technologies for use in manufacturing facilities. These facilities do not have the personnel available or the time to devote to develop software applications and hardware devices for the purpose of communicating information. This could be expensive for the company to implement and tends to be at a lower priority level for the industry as a whole. As a result, important machining process information can be “locked” in the machine for only the machine operator themselves to access during operation. Crucial information about the machining process can be lost due to the lack of communication outside of the interface between the operator and the machine tool. This information would be useful for engineers to analyze these processes to improve them or eliminate destructive conditions in order to preserve or increase part quality.

Machining conditions such as chatter can be detrimental to part quality, and the process parameters during machining must be known in order to reduce or prevent this from occurring. Chatter, or self-excited vibrations that create unstable cutting conditions [1], is an undesirable phenomenon and many machining operators attempt to avoid these conditions to preserve part quality and reduce damages to the machine tool. The prevention and suppression of chatter is usually done by experienced machine operators, and machining parameters are adjusted manually, often in a reactive manner after damage has already occurred. Machine builders are beginning to realize that automatic chatter suppression is needed in the case that an experienced machine operator is unavailable or does not detect chatter immediately, in which case part damage could have already

occurred without the machine operator's knowledge. Okuma Corporation, a builder of CNC machine tools, has developed an automatic chatter suppression system called "Machining Navi" which can be an option on any machine tool. [2] This system incorporates vibration sensors to detect chatter and adjusts the spindle speed to suppress chatter. However, this is a proprietary system which limits the availability of the data outside the machine tool controller. Additionally, any information about these destructive machining conditions from the current process would not necessarily be communicated to downstream operations that could avoid further reduction in part quality with the given information. This is of the utmost concern for manufacturers of high-value raw materials and manufacturers of hard-to-machine raw materials. High-value raw materials are expensive to replace if their surface quality was reduced during the machining process. Also, hard-to-machine materials could require replacement tooling that could be expensive and time-consuming to replace. If destructive machining conditions could be prevented while manufacturing these materials, the industry would be able to preserve and increase part quality and increase productivity in their manufacturing processes.

In addition, the tracking of part-specific information becomes unorganized if the information is lost from one machining process to the next because the process information was not identified with the part itself at the time of operation. This lack of interoperability between machining processes could decrease part quality and productivity in the manufacturing environment because of the unorganized representation of this information and the lack of communication of part-specific information to downstream operations. General Electric (GE) Power & Water in Greenville, SC

expressed a need for part-tracking capabilities in their machine tools for manufacturing gas turbine blades out of high-value materials. They realized that the part-tracking systems they previously had in place were insufficient at providing them enough important information to downstream operations and to access for further analysis of the machining processes. This communication downfall was affecting their productivity on the manufacturing line as well as in their post-analysis of the processes for improvement measures.

In the past few years, several companies have realized that these communication problems exist and in turn formed a partnership to find a solution to these problems for the industry. They developed a standardized, interoperable, and open-source format of data communication called MTConnect. The partnership was named the MTConnect Institute and worked to further develop the MTConnect standard and push for implementation of the standard in the industry. [3] The idea was to create a generalized form of data communication that would allow the proprietary data sources from machine tools to be translated into a standard format. The data would then be transmitted through common data transfer protocols and could be read by any software application for any data collection and analysis purpose. MTConnect is solely a standard of formatting information and thus requires the creation of software applications in order to use the data that would be communicated through the standard. Current software applications used on most machine tools are mostly proprietary and therefore are not interoperable between equipment types and do not represent the data in a standard format. The MTConnect Institute developed a competition called MTConnect Challenge to entice the development

of software applications that would utilize the MTConnect standard and increase the number of standardized, interoperable software applications for machine tools. Part of the motivation of this project was to create a submission for this competition to attempt to expand the usage of MTConnect and provide a solution to some of the communication issues in manufacturing.

Research Focus Areas

The focus areas of this research project are contained within the broad focus in manufacturing, as shown in a hierarchical breakdown in figure 1.1.

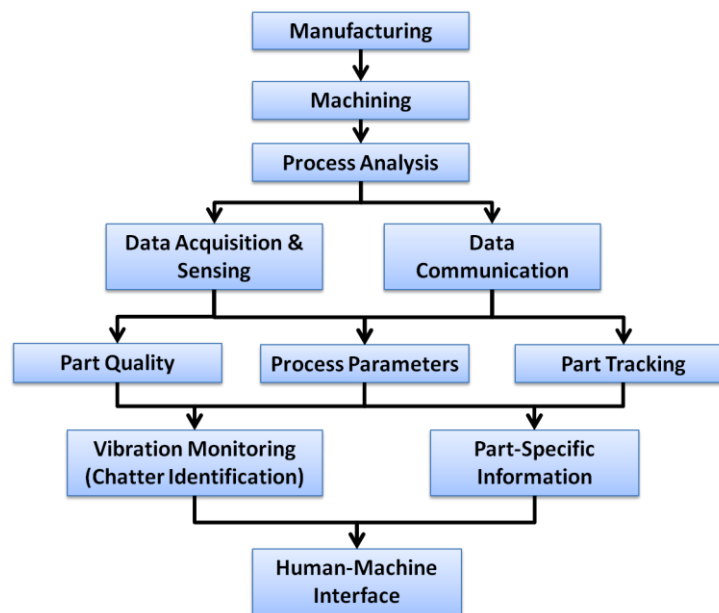


Figure 1.1 – A breakdown of the focus areas of the project.

Under the broad area of manufacturing, this research focuses on the area of machining, which is also very broad. A specific area of machining that this research

focuses on is the analysis of the machining process which involves acquiring data and sensing process phenomena as well as communicating that information. Analysis of the machining process can be done to help improve part quality, study process parameters for a better understanding of a specific machining process, and track part information for more organized data processing abilities. Part quality can be improved by monitoring vibrations to help in identifying chatter which also entails knowing certain machining parameters that are involved in the cutting process (such as spindle speed, depth of cut, feed rate, etc.). Part tracking involves knowing the process parameters for specific parts and storing that information for better data analysis and organization. A human-machine interface, such as a software application, would be the link between monitoring and communicating important data from the machining process. Therefore, this research project was developed in the specific area of manufacturing that focuses on monitoring and communicating important machining process information to help improve part quality and productivity in the industry.

Problem Statement

The project goal is to offer a solution to some of the communication, productivity, and part quality problems in the American manufacturing industry, in particular the problems with interoperability between proprietary manufacturing equipment, by providing a custom software application that integrates MTConnect, a standardized and interoperable data communication process, with proprietary data acquisition tools and custom sensors to monitor and communicate CNC machining process information to aid in chatter identification and suppression in downstream machining operations.

Objectives

The overall objective is to provide a means of data communication in machine tools that would utilize a standard format of communication along with proprietary data acquisition tools and a proprietary machine tool controller in a single custom software application to be used to monitor important machining process information in order to improve part quality. The detailed sub-objectives are the following:

1. Design a custom software application for monitoring and communicating machining process data
 - a. Receive data from the machine tool controller using a standard format of communication: MTConnect
 - b. Receive data from custom sensors via proprietary data acquisition (DAQ) hardware and software
 - c. Monitor remotely using an intuitive and easy-to-operate user interface
 - d. Track part-specific information to preserve data integrity and for future analysis
2. Design a custom software application for improving operator input to part programs and control
 - a. Create an input approach that is easy to use and more accurate than operator typing
 - b. Communicate input data using the interoperability standard
3. Enter the MTConnect Challenge to demonstrate application functionality

4. Implement the application on a machine tool
 - a. Identify chatter for increased part quality and productivity
 - b. Demonstrate how it can be integrated directly to a machine tool controller

Organization

The remainder of this thesis is organized as follows: Chapter 2 provides a background of seminal and critical works related to interoperability of manufacturing processes and systems, as well as previous quality improvement approaches to process control. Chapter 3 describes the specifications of the designed system, including hardware and software systems integrated to the application. Chapter 4 identifies the software development tools that were used in creating the application, and the architecture of each component and major functional area of the application. Chapter 5 describes the system's data communication process and the operational procedure of the entire system and the application's user-interface. Chapter 6 details the implementation of the application for use in chatter identification. Chapter 7 provides a list of the limitations and shortcomings of the technologies used in the application, a recommendation of future work to be done based on the implementation vision of the application, and several benefits to the manufacturing industry. Chapter 8 is a summary of the research project and conclusion of the thesis.

CHAPTER TWO

BACKGROUND

A review of the literature and work in the area of communication and interoperability in manufacturing is presented, particularly a brief history of the development of communication technologies in manufacturing and the work leading into the ideation and development of MTConnect. This chapter will also present information about what the MTConnect standard is, how it was developed, and for what purpose, as well as a description of the elements of the standard. Also, information about the structure of the MTConnect communication process will be presented as well as how it can be used to communicate data from any machine tool in a standardized format. The purpose and requirements of the MTConnect Challenge will be discussed. Additionally, a review and critique of related work using MTConnect will be presented.

Also in this chapter, the importance of vibration monitoring in industrial manufacturing will be discussed, and some previous work on identification, prediction, and suppression of chatter will be presented in order to detail what has been done in this research area and discuss how MTConnect could benefit their research in this area. Some data acquisition techniques using the National Instruments CompactRIO hardware and LabVIEW software will be discussed to present the capabilities of the equipment for monitoring and control applications.

History of Communication in Manufacturing

In America, the Industrial Revolution was a key factor in launching the manufacturing industry that exists today. Many small companies were created during this period allowing the manufacturing of goods in this country to increase significantly. It was relatively easy for the workers in these small companies to communicate with each other about the manufacturing processes and the products they were selling to customers, even if it was simply on a verbal level. Over time, as the mass production of goods became the norm in manufacturing, these companies grew in size, merged with others, and began to form different departments within the companies. These departments became isolated from one another, straining the communication between members of the organization about the products they were manufacturing. This was an issue that was difficult to overcome until the invention of the digital computer in the 1950's, which introduced the idea of automation, control, and information management of manufacturing processes as a possibility for the industry. The importance of process "intelligence" and databases of process information was understood by craftsmen and engineers, but at that point the information was only able to exist in the mind of the operator of that process or on the paper in which it was written, a problem that still exists in varying degrees today. The idea of actually implementing computers into the manufacturing industry was not accepted until after the computer was proven to be an extremely powerful tool for understanding manufacturing as a system, during the 1960's. [12] However, the computer was still in its infancy and was expensive, requiring enormous rooms filled with equipment and large rolls of tape for each computer program.

The late 1960's saw the development of computers that were physically smaller and had faster processing capabilities, and were beginning to be implemented in manufacturing facilities and small businesses to improve data processing. [51]

During the 1970's and 1980's, Computer Integrated Manufacturing (CIM) systems were being developed and implemented to automate and optimize manufacturing processes. These computer systems were linked on a central network, helping to introduce the idea of networked communication within manufacturing systems. There were many benefits including improved productivity and quality, decreased costs, and increased worker and customer satisfaction. However, initially only a few companies were able to realize these benefits because of the expense and problems with utilizing these technologies to their full potential. In the 1980's and 1990's, studies were performed to analyze these companies on how well they utilized the available technologies and found that an important component of the successful implementation of these CIM systems is the communication of ideas and data between personnel [83], [84]. It was found that the majority of the machine tool utilization time was spent unproductively, between "waiting and moving", "positioning, loading/unloading, gaging, etc." and only a very small amount of time was spent on cutting parts on the machine tool. This led to the development of Numerically Controlled (NC) machine tools which allowed these processes to become more efficient and productive, as well as led to the ideas behind "lean manufacturing". A new question that came out of the development of NC machine tools and the need for data communication of manufacturing processes was

how technologies such as computers and networks of computers could help meet these needs. [12]

In the 1980's and 1990's, research was being performed on various machining processes using telemetry systems on manual machine tools and utilizing sensor information to aid in understanding the machining process. These experiments relied on hard-wired circuits to sense physical phenomena in the machining process and convert the signals to a frequency modulated (FM) signal to be transmitted to a data acquisition unit on a computer [52], [13]. As computing technology evolved into the 2000's, the data communication process with machine tools began to utilize standard digital communication hardware such as RS-232 serial connections instead of radio wave transmitters and receivers. [14] Many machine tools in industrial manufacturing facilities still use RS-232 connections today, but are transitioning to more efficient and standardized hardware connections such as Universal Serial Bus (USB) to connect to peripheral equipment. Also, modern machines are being equipped with Ethernet capabilities to be able to connect to internal networks and the Internet to improve data communications and remote monitoring of machining processes. However, the transition to these modern machines is slow due to the fact that the "legacy" machines, or older machines that have RS-232 connections, are still functioning properly and manufacturing facilities do not have the need to replace these with more modern machines which can be expensive [14].

As the need for better data communication methods in manufacturing becomes more apparent, the industry will continue moving closer toward being a "digital factory",

or a system of technologies that enable full communication and cooperation between entities of a manufacturing system [12]. This should help convince manufacturing facilities to transition to more modern machine tools with better data communication capabilities, including those with open-architecture systems. These types of systems allow open access by any user to components of the machine tool controller to aid in collecting data and adjusting process parameters on-line. Some of the important principles behind the “digital factory” include standardization, data integration, work flow management, and automation of planning, with software being the connection between humans and machining models, intelligence, processes, and design. [12] Since many manufacturing facilities have hundreds, if not thousands, of machines on the floor, each with a different method of data communication, integrating them and connecting them to a common network or software monitoring application has proven to be difficult. This led to the development of standardized data communication methods between machine tools in order to work toward the “digital factory” goals of standardization and data integration, such as MTConnect. [18]

System Interoperability

Interoperability of systems is a key to providing seamless data flow and transparency, and leads to greater opportunities within manufacturing systems for leveraging information generated during the manufacturing process to improve quality and productivity. This goal of “the data you need, when you need it, where you need it” is a principle that guides how interoperability of manufacturing systems is being pursued. The latest major effort is the MTConnect standard being piloted through the Association

for Manufacturing Technology, a nonprofit group with the mission of improving manufacturing viability in the US. Because of this standard, and other similar standards, different proprietary manufacturing systems are now able to “talk” to each other in the same “language”. For example, a solution to the interoperability problems of common industrial robots and machine tools was created by the National Center for Defense Manufacturing and Machining (NCDMM) and the National Institute of Standards and Technology (NIST) with an application that would bridge the gap between these two systems [79]. Common industrial robots use software called ROS-Industrial which is normally incompatible with the software on machine tools. The application translates the data from the robot software in order for it to communicate with MTConnect-compliant machine tools. This allows the robot to load and unload parts into the machine tool precisely when it is ready to accept new parts [79]. This is a task that would be difficult and complex without the assistance of interoperable applications using standards such as MTConnect.

Other efforts toward system interoperability have also been made through NIST. A standard called STEP-NC was created to connect Computer-Aided Design (CAD) and Computer-Aided Manufacturing (CAM) data on machine tools [76], [14]. This was designed to aid in the creation of “smart applications” that would recognize the design requirements from the CAD program and automatically translate that information to the manufacturing process requirements to allow interoperability between these otherwise separate software systems.

Several interoperability issues have been recognized also in manufacturing quality assurance, which typically involves inspecting parts using equipment and computer software from various sources. Solutions to this problem consist of standards such as the I++ Dimensional Measuring Equipment (DME) [77] and projects like the Quality Information Framework (QIF) [82] that would allow quality measurement systems to communicate with each other in a common language and aid in the efficiency of part inspections.

There are also interoperability issues in the communication of data between the shop floor and machining models and simulations due to the “language barrier” between these systems. This is preventing the machining models and simulations from gathering the appropriate data from the actual machining processes for analysis from the lack of standard data representations [78]. Partners at NIST have developed the Core Manufacturing Simulation Data (CMSD) as part of the solution to this problem in data communication between the shop floor and computer simulations. [80]

The interaction of data between system components is necessary for the “digital factory” to be completely interoperable. This involves standardizing the communication process between these systems to develop a common “language” that all manufacturing systems can “speak”. If all of the systems speak the same language along with the applications that aid in the data collection and analysis processes, the hardware and software components of these systems will become plug-and-play. This plug-and-play interoperability between systems is similar to how one could plug in their USB flash drive and their computer instinctively speaks the “language” of the USB drive to be able

to connect, store, and retrieve data. This type of natural communication between systems is envisioned to become the future of information transfer in manufacturing with standards such as MTConnect.

Introduction to MTConnect

MTConnect is an open-source communication standard developed to exchange data. It is supported by industry and academic partners in the MTConnect Institute. [3] and was introduced in 2008 [18]. Together, the partners realized a need for an interoperable, standardized way of communicating data from proprietary manufacturing equipment.

Each piece of equipment, developed by a specific machine builder, has its own proprietary communication method in its controller to collect, organize, and display data from the machining process. In order for a software application to access this data, it would need to convert each piece of data from its proprietary format, which differs greatly from one machine to the next. MTConnect is a solution that increases interoperability between devices by providing a common format in which data is communicated from the machine controller. This allows any software application to have access to the data from the machine controller without going through the process of translating the proprietary data in the application itself. It also gives manufacturing facilities the opportunity to use the same application on each of their machine tools, regardless of what type of equipment it is or what machine builder it came from. This reduces the need for application development for each individual machine tool, reduced time for training on different systems, improved consistency of operation and data

interpretation, and a greater interoperability between devices on a manufacturing floor. The ability to remotely monitor multiple machining processes at once and the ability to have a greater understanding of each machining process can be easily achieved through MTConnect. [18]

The main feature of the MTConnect standard is that it utilizes the Extensible Markup Language (XML) which is a common language used in transporting and storing data in a structured format with the ability to customize and extend the language capabilities. [4] The XML schema is the format in which data is stored and can be set up in any configuration depending on the type of data to be transmitted. A software application can be developed to send, receive, and display this data. The benefit of using this language for the MTConnect standard is that it is human and machine readable, customizable for any data representation, and it is independent of any hardware or software platform. It is also widely used across many technologies, particularly in sharing data over the Internet. [4] The MTConnect standard uses the XML standard to represent data from the machine tool according to specified data categories, and transmits the data across common data transfer protocols. Data is transmitted using the Hypertext Transfer Protocol (HTTP) with the Transmission Control Protocol, a type of Internet Protocol (TCP/IP), which are the fundamental data transfer protocols in the creation of the World Wide Web. This gives the MTConnect the ability to become a standard format of data communication on machine tools, most of which natively have access to these data transfer protocols. This also gives MTConnect “plug-and-play” interoperability between devices due to these common data transfer protocols, meaning that it is easy to install

with limited setup procedures and applications can simply “plug into it” and have access to the information without any additional procedures.

Other important features about MTConnect are that it is open-source which allows anyone access to the standard, its associated documentation, and example software packages that help initialize the usage of MTConnect in any manufacturing environment. The standard and all related software packages are royalty-free, meaning that they do not require any licence fees. The standard is also open to modification by anyone, provided the revisions are reviewed by the MTConnect Institute and deemed appropriate for implementation into the standard. This is helpful for the standard to continue to grow and adapt to different needs of the industry over time.

MTConnect Communication Process

The MTConnect standard simply defines the way manufacturing data is formatted and presented to systems. The entire communication process involves some equipment and software that is not specifically defined by the MTConnect standard but is important in communicating the information from the machine tool. The schematic in figure 2.1 shows the most basic architecture of the MTConnect communication process defined by the MTConnect Institute [5].

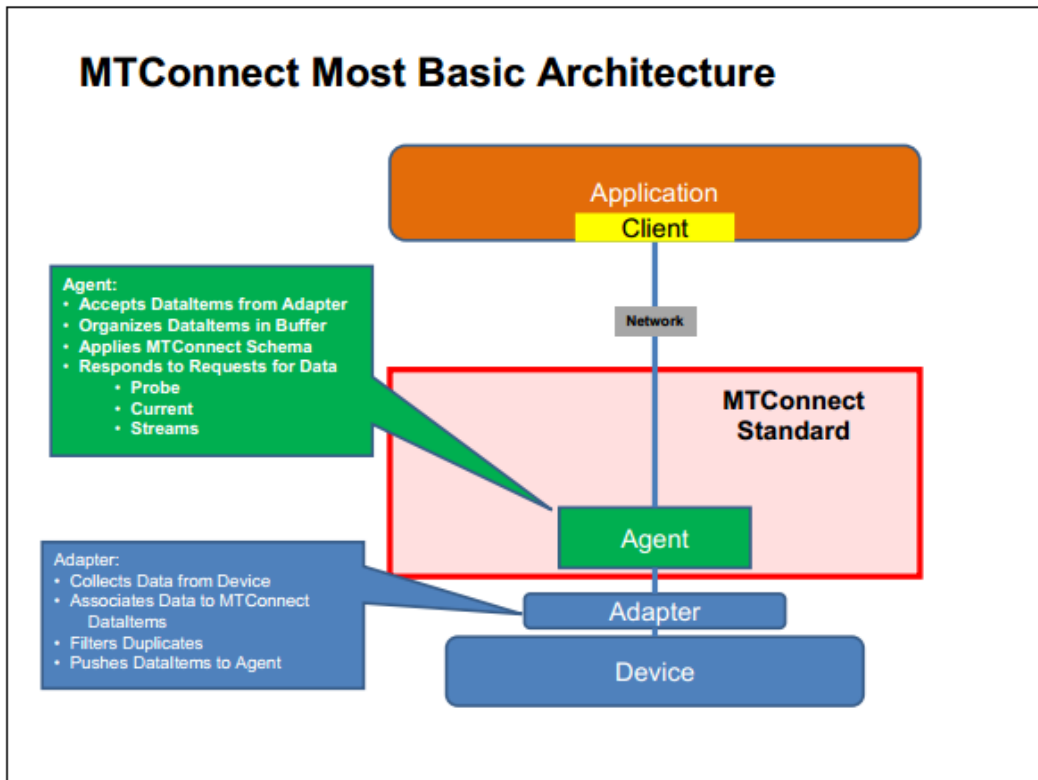


Figure 2.1 – The basic architecture of an ideal MTConnect system. [5]

The process starts with the *device* which is defined as any machine tool or other manufacturing process equipment that has a data source of some kind, usually from a controller. The device contains the data source in its proprietary format. Depending on the type of machine tool and whether or not it has MTConnect capabilities natively installed, an *adapter* may be required to translate the proprietary data format into the MTConnect language. For most modern machines, a software adapter would perform these duties, but for older machines with limited computing power a hardware adapter would be required. Specifically, the adapter collects the proprietary data from the device and associates it to the appropriate MTConnect-defined *DataItem*. A *DataItem* is the

descriptive information of a certain *component*, which is a part of the device such as the X-axis of the tool on the machine. For example, a *DataItem* of the X-axis component could be the position of the X-axis of the tool. Once the data is translated into appropriate *DataItems* that the MTConnect standard supports, the adapter pushes the *DataItem* streams to the *agent*. [5], [6]

The *agent* is a piece of software that collects, organizes, and arranges the data from the adapter (or directly from the device if no adapter is required, such as on an MTConnect-native device) according to the schema defined by the MTConnect standard. The agent is the only part of the process that is specifically defined by the MTConnect standard. All of the other pieces of the process are there to support the transfer of data to and from the agent. Specifically, the agent accepts the *DataItems* from the adapter, stores them in a buffer, arranges them in the standard schema, and receives and processes requests from *applications* to transmit the required data. The *application* defines the software that consumes the data in order to store, manipulate, and display it to the user on an interface. The application usually includes a *client* which is a software function that connects to the agent and sends the requests for retrieving data [5], [6]. The data is sent over a network, which typically involves an Ethernet connection or wireless connection to the Internet or intranet (internal network) using standard communication methods such as HTTP and TCP/IP.

The requests that can be sent to the agent use Representational State Transfer (REST) which is an architecture that defines how data moves between the client and the web server based on commands that are given [6], [7]. The commands that can be given

to the MTConnect agent are “Probe”, “Current”, and “Sample”. The “Probe” command is used when the properties and configurations of the device itself are required. The “Current” command is used to retrieve a “snapshot” of the current values of all of the DataItems under each component of the device, or individual DataItems that are specified with the command. The “Sample” command shows a stream of data points in a certain time period of all of the DataItems under each component, unless individual DataItems are specified with the command [6].

The MTConnect Institute set up an agent that interfaces with a machine tool simulator to provide users with a sample agent to reference when either creating their own agents or debugging their applications. Anyone can access the sample agent by entering the agent address into a web browser. The agent address is typically a URI (Universal Resource Identifier) that is commonly known as a web address [6]. The user would enter “www.agent.mtconnect.org” into a web browser and be able to view the MTConnect schema in its XML-based format. Figures 2.2, 2.3, and 2.4 show the MTConnect schema when the following commands are sent to the sample agent: probe, current, and sample, respectively.

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<MTConnectDevices xmlns:m="urn:mtconnect.org:MTConnectDevices:1.2"
xmlns="urn:mtconnect.org:MTConnectDevices:1.2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="urn:mtconnect.org:MTConnectDevices:1.2
http://www.mtconnect.org/schemas/MTConnectDevices_1.2.xsd">
  <Header creationTime="2014-02-17T21:00:55Z" sender="mtconnect" instanceId="1378432262"
  version="1.2.0.21" assetBufferSize="1024" assetCount="0" bufferSize="131072"/>
  ▼<Devices>
    ▼<Device id="dev" iso841Class="6" name="VMC-3Axis" sampleInterval="10" uuid="000">
      <Description manufacturer="SystemInsights"/>
      ▼<DataItems>
        <DataItem category="EVENT" id="avail" type="AVAILABILITY"/>
        <DataItem category="EVENT" id="dev_asset_chg" type="ASSET_CHANGED"/>
      </DataItems>
      ▼<Components>
        ▼<Axes id="ax" name="Axes">
          ▼<Components>
            ▼<Rotary id="c1" name="C">
              ▼<DataItems>
                ▼<DataItem category="SAMPLE" id="c2" name="Sspeed" nativeUnits="REVOLUTION/MINUTE"
                subType="ACTUAL" type="SPINDLE_SPEED" units="REVOLUTION/MINUTE">
                  <Source>spindle speed</Source>
                </DataItem>
              </DataItems>
            </Rotary>
          </Components>
        </Axes>
      </Components>
    </Device>
  </Devices>
</MTConnectDevices>
```

Figure 2.2 – The MTConnect schema when the “Probe” command is sent to the sample agent at “http://agent.mtconnect.org”. The red box shows an example of a component, the rotary “C” axis, and one of its associated DataItems, which is defined as the “spindle speed” with the units of “revolution/minute”.

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<MTConnectStreams xmlns:m="urn:mtconnect.org:MTConnectStreams:1.2"
xmlns="urn:mtconnect.org:MTConnectStreams:1.2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="urn:mtconnect.org:MTConnectStreams:1.2
http://www.mtconnect.org/schemas/MTConnectStreams_1.2.xsd">
  <Header creationTime="2014-02-17T21:04:01Z" sender="mtconnect" instanceId="1378432262"
  version="1.2.0.21" bufferSize="131072" nextSequence="2791825357" firstSequence="2791694285"
  lastSequence="2791825356"/>
  ▼<Streams>
    ▼<DeviceStream name="VMC-3Axis" uid="000">
      ▼<ComponentStream component="Rotary" name="C" componentId="c1">
        ▼<Samples>
          <SpindleSpeed dataItemId="c2" timestamp="2014-02-17T21:01:39.550300" name="Sspeed"
          sequence="2791790385" subType="ACTUAL">3400.0000000000</SpindleSpeed>
          <SpindleSpeed dataItemId="c3" timestamp="2014-02-17T20:57:37.064156" name="Sovr"
          sequence="2791735008" subType="OVERRIDE">100.0000000000</SpindleSpeed>
          <Load dataItemId="cl3" timestamp="2013-09-06T01:51:02.572398Z" name="Cload"
          sequence="15">UNAVAILABLE</Load>
        </Samples>
        ▼<Events>
          <RotaryMode dataItemId="cm" timestamp="2013-09-06T01:51:02.572398Z" name="Cmode"
          sequence="18">SPINDLE</RotaryMode>
        </Events>
        ▼<Condition>
          <Normal dataItemId="Cloadc" timestamp="2014-02-17T20:57:37.064156" sequence="2791735015"
          type="LOAD"/>
          <Unavailable dataItemId="Csystem" timestamp="2013-09-06T01:51:02.572398Z" sequence="2"
          type="SYSTEM"/>
        </Condition>
      </ComponentStream>
    </DeviceStream>
  </Streams>
</MTConnectStreams>
```

Figure 2.3 – The MTConnect schema when the “Current” command is sent to the sample agent at “http://agent.mtconnect.org”. The green box shows a “ComponentStream”, which is a stream of data from a component, of the rotary “C” axis and the corresponding values of a “Sample” defined as the “Spindle Speed” DataItem. The red box highlights the current value of the actual spindle speed.

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<MTConnectStreams xmlns:m="urn:mtconnect.org:MTConnectStreams:1.2"
xmlns="urn:mtconnect.org:MTConnectStreams:1.2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="urn:mtconnect.org:MTConnectStreams:1.2
http://www.mtconnect.org/schemas/MTConnectStreams_1.2.xsd">
  <Header creationTime="2014-02-17T21:03:00Z" sender="mtconnect" instanceId="1378432262"
  version="1.2.0.21" bufferSize="131072" nextSequence="2791680103" firstSequence="2791680003"
  lastSequence="27916811074"/>
  <Streams>
    ▼<DeviceStream name="VMC-3Axis" uuid="000">
      ▼<ComponentStream component="Path" name="path" componentId="pth">
        ▼<Samples>
          <PathFeedrate dataItemId="Frt" timestamp="2014-02-17T20:51:00.097743"
          sequence="2791680019">0.3736286</PathFeedrate>
          <PathFeedrate dataItemId="Frt" timestamp="2014-02-17T20:51:00.113742"
          sequence="2791680024">0.2531968</PathFeedrate>
          <PathFeedrate dataItemId="Frt" timestamp="2014-02-17T20:51:00.141742"
          sequence="2791680035">0.3612953</PathFeedrate>
          <PathFeedrate dataItemId="Frt" timestamp="2014-02-17T20:51:00.169741"
          sequence="2791680040">0.4</PathFeedrate>
        </Samples>
      </ComponentStream>
    </DeviceStream>
  </Streams>
  ▼<Events>
    <Block dataItemId="cn2" timestamp="2014-02-17T20:51:00.129742" name="block"
    sequence="2791680030">X-0.972157 Y-1.091950</Block>
    <Line dataItemId="cn4" timestamp="2014-02-17T20:51:00.129742" name="line"
    sequence="2791680025">276</Line>
  </Events>
```

Figure 2.4 – The MTConnect schema when the “Sample” command is sent to the sample agent at “http://agent.mtconnect.org”. The green box shows the “ComponentStream” of the “Path” component, and the red box shows the stream of samples of the “Frt”, or feed-rate, DataItem with associated values along with a timestamp to show the time at which the agent was sampled to retrieve this data.

Through the MTConnect communication process, data can be retrieved from a device in “near real-time”. There is a slight delay (about 1 second) between the adapter and the agent to retrieve data, which is then buffered and stored in the agent for a certain length of time, and a short delay between the agent and the application depending on the processing speed of the PC that hosts the application and how fast the application can connect to a network. However, the delays are not significant enough to hinder the abilities of MTConnect to become a primary source of data communication in the manufacturing industry.

Previous Work with MTConnect

Since the introduction of MTConnect, several implementations have been executed to discover the feasibility and functionalities of the standard. These systems are able to connect to the machine tool and gather information that would normally be “locked” in the controller. Some implementations include process planning [21, 18], tool position analysis [18, 26], feed-rate analysis [18], all which utilize the information from the machine tool through the MTConnect standard to perform some analysis that would normally require complex data translation from proprietary formats. Other implementations use MTConnect to monitor the machining process on-line. Such systems include an energy consumption monitoring system that is used to optimize the performance of the machining process [22] and a Machine Tool Information Service System (MTISS) which uses MTConnect to collect data on machine tools to be sent to remote hosts for monitoring and analysis [23]. Also, the “WatchDog Agent” was created to monitor the health of a machine tool and provide information that would help decrease maintenance requirements. This implementation also verified the capabilities of MTConnect in connecting to a machine tool controller and interfacing with LabVIEW, a graphical programming software package for data acquisition [26].

More computationally advanced systems that have been implemented using MTConnect include an event-based real-time temperature control system which uses the information gathered through inter-connected MTConnect devices as feedback for its control algorithm [25]. Also, a system was designed that utilizing actual shop-floor information from MTConnect to update kinematic machining models based on the STEP-

NC standard that translates CAD data to CAM data [24]. This process would be complex without the assistance of both MTConnect and STEP-NC.

Several systems using MTConnect have been designed with a user-interface in mind. For example, the “Probe Direction Analyzer” [26] was developed in the Java programming language with a user-interface as seen in figure 2.5. This application was designed to increase the accuracy of on-machine probing by providing information about the probe’s position through MTConnect to aid in calculating the movement of the probe in certain specified directions and provide the errors in positions to the user.

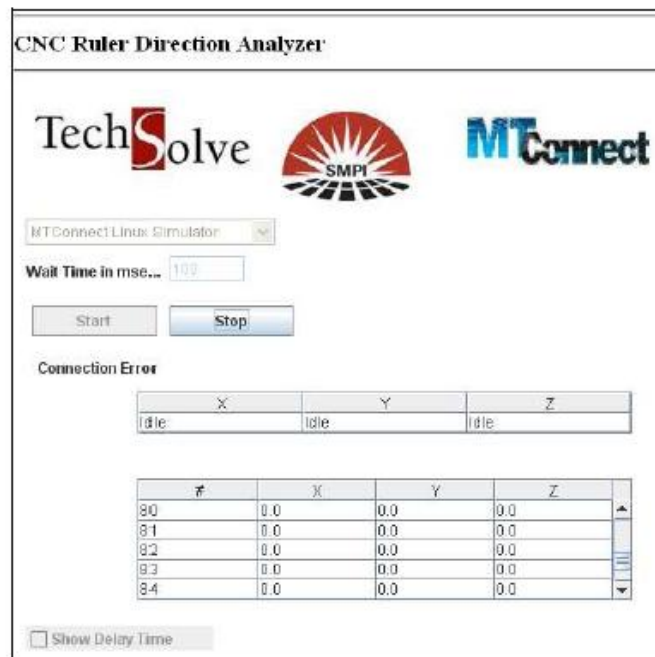


Figure 2.5 – The user interface of the “Probe Direction Analyzer” application. [26]

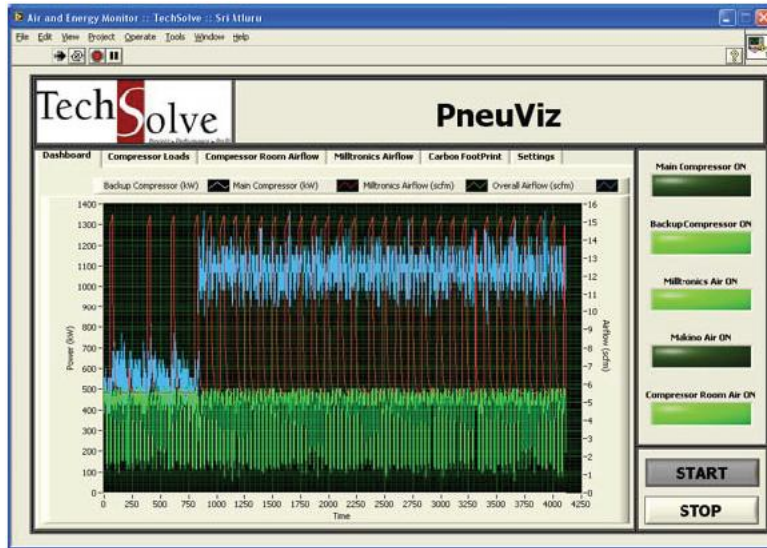


Figure 2.6 – The user interface of the “PneuViz” application. [27]

The “Pneuviz” application [27] provided a user interface in LabVIEW for monitoring the compressed air used in machining processes in order to optimize the efficiency of the compressed air system. This interface is shown in figure 2.6. Also, a web-based quality inspection monitoring application was programmed in C# and designed using Excel as the user-interface [28]. This application displayed updated inspection results to the user to provide quality feedback as seen in figure 2.7.

The screenshot shows an Excel spreadsheet titled 'Quality Monitoring'. It contains a data table with the following columns: Date, Part, Characteristics, Feature, Status, Target, Actual, Lower Tolerance, Upper Tolerance, and Error. The data rows show inspection results for different parts and features, with status indicators in green and red.

Date	Part	Characteristics	Feature	Status	Target	Actual	Lower Tolerance	Upper Tolerance	Error
8/13/2011 13:27:52		DiameterCharacteristicsDistance	CrDrReasurement3283	GO	44.0000	44.0002	44.0000	44.0000	0.0002
8/13/2011 14:27:53		DiameterCharacteristicsDistance	CrDrReasurement3284	GO	44.0000	44.0004	44.0000	44.0000	0.0004
8/13/2011 15:27:55		DiameterCharacteristicsDistance	CrDrReasurement3285	NO	44.0000	43.9998	44.0000	44.0000	0.0004
8/13/2011 16:27:55		DiameterCharacteristicsDistance	CrDrReasurement3286	NO	44.0000	43.9998	44.0000	44.0000	0.0002

Figure 2.7 – The user interface of the “Quality Monitoring” application. [28]

These applications were successful in implementing MTConnect to provide data to the user that would previously have been complex or impossible to retrieve. Because MTConnect is a relatively new standard, there are few implementations that include software applications that have easy-to use, intuitive user-interfaces for displaying information gathered through MTConnect.

MTConnect Challenge

The MTConnect Institute has realized the potential for this communication process to revolutionize the manufacturing industry. However, there are currently a limited number of software applications available for monitoring machining operations using MTConnect. Manufacturing facilities themselves usually do not have the time or resources to develop these types of applications. As a result, the MTConnect Institute developed the MTConnect Challenge, supported and funded by the U.S. Department of Defense, to entice software developers to create these applications to expand the usage of MTConnect in the industry.

The MTConnect Challenge is a two-part competition with cash prizes to the top winners with the best ideas or applications that meet the competition requirements. The first part, Challenge 1 [9], is a proposal of innovative ideas using MTConnect. The second part, Challenge 2 [10], is the implementation of innovating and breakthrough software applications using MTConnect. The top 5 winners of Challenge 1 were identified before the submission of Challenge 2 began. Submitting to Challenge 1 was not a prerequisite for submitting to Challenge 2, but helped to organize ideas for innovative applications using MTConnect.

Challenge 2 consists of three phases. The first phase selected the top 10 best applications out of all of the paper submissions. The second phase of judging selected the top 5 applications after a web conference with the judges. The top 5 submitters would then be required to present at the annual MTConnect: Connecting Manufacturing Conference [11], known as [MC]² or MC2. The third phase would select the top 3 winners after the presentations at the conference. The top 3 winners would receive cash prizes.

The original idea behind this application was submitted to Challenge 1 and the actual application that was developed was submitted to Challenge 2. During Challenge 2, the application was selected as a top 10 finalist in the first phase and a top 5 finalist in the second phase of judging. For the third phase of judging, a live demonstration was presented at the MC2 conference in Orlando, FL on April 9th, 2014. The selection of the top 3 finalists occurred after the conference attendees voted for the top 3 applications to win awards. This competition series helped motivate individuals or companies to develop software applications using MTConnect that would hopefully be implemented in manufacturing facilities and help to expand the usage of MTConnect.

Part Quality Preservation

Vibration Monitoring

Industrial manufacturing companies are interested in vibration monitoring to help preserve the quality of the parts in production and machining productivity. Vibration monitoring is a subset of condition monitoring as a part of predictive maintenance strategies. This involves monitoring the condition of machinery during operation and analyzing the information to predict the failure modes of the components of the machine [41]. Monitoring the vibrations of rotating machinery is important in detecting any destructive conditions before failure occurs instead of waiting until after the machine components fail or performing maintenance on components that still have usable life left. This can also be translated to vibration monitoring of the machining process itself, where the early detection of destructive conditions due to unstable vibrations would reduce the cost and amount of replacement parts which would be needed after surface quality is compromised.

Vibration monitoring can be categorized into detection, diagnosis, and prognosis. Detection involves measuring vibration levels, commonly in a range of frequencies. These measurements can then be used in studies that would analyze the process and study the deterioration of certain conditions and failures that could have been caused by excessive vibrations. This is often shown by plotting a trend of vibration levels over time or compared with publications of common vibration levels of similar components which would indicate the development of deteriorations and diagnose the failure modes. Then, the remaining useful life of the component and its possible failure modes can be

identified in the prognosis stage [41]. This can lead to adjusting parameters for subsequent machining operations that would eliminate or reduce the destructive vibrations to preserve part quality and the life of the machine itself.

Chatter Identification and Prediction Methods

Chatter is the self-excited vibrations of the interaction between the cutting tool and the workpiece during a machining operation [1]. Since the cutting tool is not perfectly rigid and has a certain stiffness and mass associated with it, it is allowed to deflect during cutting. This causes the tool to vibrate while removing material which leaves a “wavy profile” on the surface of the workpiece [1], as shown in figure 2.8.

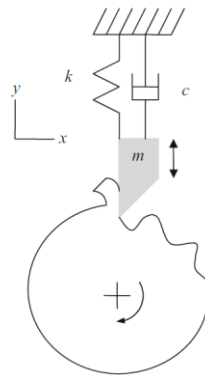


Figure 2.8 – A schematic of the “wavy profile” generated by a vibrating tool. [1].

In the turning process on a lathe, subsequent revolutions of the workpiece will introduce the cutting tool to this “wavy profile” which will vary the chip thickness and will therefore vary the cutting force proportionally. The “regeneration of waviness” will

cause chatter if the waves between revolutions are out of phase because of a larger variation in chip thickness [1]. This is demonstrated in figure 2.9.

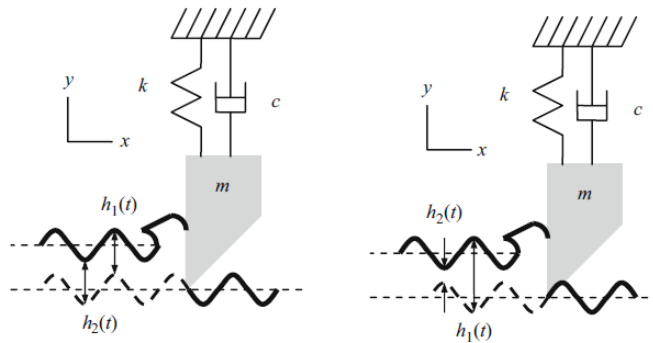


Figure 2.9 – In-phase wavy profiles generates a negligible chip thickness from one revolution to the next (left) and out-of-phase wavy profiles generate a variable chip thickness and therefore variable cutting force from one revolution to the next (right). [1]

Several studies have been done on the early prediction of chatter in both turning and milling processes. This typically involves analyzing the cutting forces on the machine tool to model and predict chatter dynamics [35], [36], [37]. Various sensors have been used to detect cutting force variations and vibrations in these experiments and the results have been used to update dynamic models of the process to help predict the pattern of the vibrations [38]. Additionally, stability lobes diagrams have been developed based on these models and used to predict chatter and select cutting parameters that would help lead to stable cutting conditions [1], [39]. Stability lobe diagrams show the range of values of the limiting chip width against the spindle speed for specific machining processes that would lead to stable and unstable conditions. This is commonly used to identify if a specific process would be stable or unstable based on these

parameters. Figure 2.10 shows a simplified stability lobe diagram with stable and unstable regions highlighted.

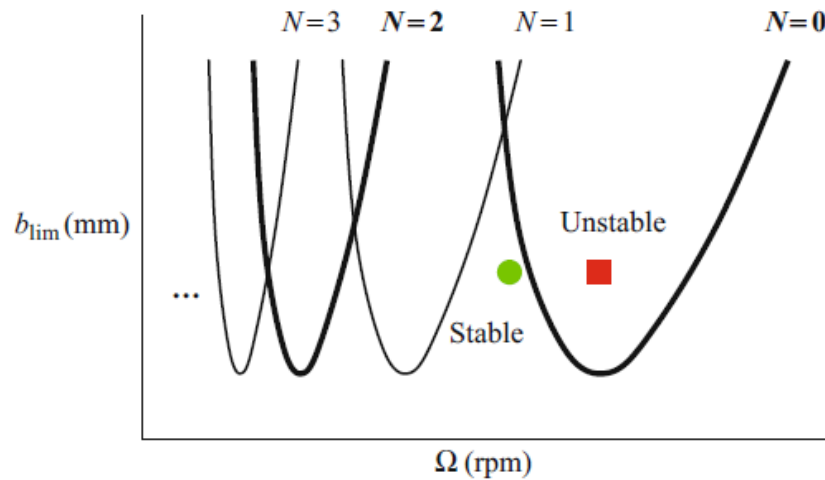


Figure 2.10 – A simplified representation of a typical stability lobe diagram. [1]

The stability lobe diagram can be useful when the machining model and the frequency response function of the system are known. However, in a full production manufacturing facility, the machine operators may not know the models, frequency response functions, and parameters like the limiting chip width for each machine, and they may not have the time to perform these calculations to keep on the production schedule. A simpler, more practical method of chatter identification using live sensor data would be more beneficial in industrial manufacturing.

CHAPTER THREE

MACHINE, HARDWARE, AND SOFTWARE SPECIFICATIONS

The system configuration was developed to meet the objectives of the project: to design a custom software application to integrate data communication through MTConnect and proprietary data acquisition tools and custom sensors to monitor and communicate machining process information to aid in chatter identification. In order to accomplish the objectives, data acquisition (DAQ) hardware and associated software were set up on a host PC by the machine tool to connect to the custom sensors integrated on the tool, and the MTConnect components were installed on the machine's PC which could then be accessed by the application on the host PC over a network. The following sections will discuss the details of the CNC machine, the custom integrated tooling sensors, the DAQ hardware and software, and the architecture and MTConnect setup that was involved in developing and operating this system.

CNC Machine

The machine tool used in this system is an Okuma SPACETURN LB4000EX horizontal 2-axis CNC (Computer Numerical Control) lathe with THINC-OSP control (version OSP-P200L), shown in figure 3.1. Okuma's THINC is an Intelligent Numerical Control (INC), PC-based controller with a Microsoft Windows-based operating system. This allows it to be an open-architecture system that is compatible with various non-proprietary tools and applications. The controller also has multiple USB ports to connect

to various hardware devices and standard Ethernet capabilities to connect to a network or access the Internet [55], [56].

The LB4000EX has a high power, high torque Primary coil Excited (PREX) synchronous motor, which is a permanent magnet interpolation reluctance motor that generates torque through magnetic reluctance. This allows the motor to be more efficient than the inductance motors traditionally used in spindles [53], [54], [55]. This particular lathe can provide high quality machining of heavy and high-speed cutting and can accommodate larger workpieces. There are several “Intelligent Technologies” available as options on this machine including Machining Navi, a chatter suppression system, and the Collision Avoidance System. Each LB4000EX comes standard with the Thermal-Friendly Concept, which ensures there is minimal thermal growth of the machine components through the cutting process [55].



Figure 3.1 – Okuma SPACE TURN LB4000EX 2-axis horizontal CNC lathe with THINC OSP-P200L controller. [55]

The Clemson University International Center for Automotive Research (CU-ICAR) facility has a unique relationship with Okuma where researchers are given certain machining centers to perform experiments. The particular LB4000EX in the laboratory at CU-ICAR was equipped with the standard Thermal-Friendly Concept but neither of the optional “Intelligent Technologies”.

Custom Integrated Tooling Sensors

A former student at CU-ICAR developed a custom integrated tooling force sensor to be used in various experiments with force and tool wear on the machine [59]. This sensor was developed for data acquisition and control applications using force data from the machine tool, and was demonstrated for use with chatter identification, tool wear indication, and adaptive cutting force control in machining [49], [59]. The sensor is comprised of Vishay 062UV strain gauges on all four faces of the cutting tool in shear configuration that can be related to the cutting and feed force directions. The raw voltage signal from the strain gauges is amplified through two single-channel DC transducer amplifiers from RDP Electronics Ltd [85]. The initial calibration procedure involved using a load cell capable of measuring 5000 N of force to set the amplifier gain where 1 mV output is equivalent to 1 N of force. In addition to the initial calibration procedure, the zero setting on the amplifier must be set so that the signal reads approximately 0 V after the tool is secured into the tool holder on the machine (since it will be pre-loaded after being secured). After testing was performed on the sensor to determine its functionality, it was determined that small variations in the hydraulic pressure of the machine’s components could be registered as micro-strain and then amplified as force,

which is the primary source of variation in the strain gauge readings that were collected, showing a standard deviation of about 8N [59]. Also, some thermal effects of cutting were demonstrated while operating in dry conditions (no coolant). A thermocouple was fitted into the tip of the tool to mitigate these effects, where a linear model in the data acquisition software could be fit to counter the drift in voltage due to thermal effects.

Because a thermocouple was integrated into the tool as well, the application takes advantage of its availability to be able to monitor the tool temperature as well as the force variations during machining. The thermocouple is of Type K (Nickel-Chromium, Nickel-Aluminum) from Omega [58]. The calibration procedure for the thermocouple is performed on the software level after the raw voltage readings are required through the data acquisition hardware. The raw voltage signal is fit to a linear relationship between voltage and temperature, which is pre-defined in the data acquisition software.

Data Acquisition Hardware

The hardware that was used to acquire the signals from the integrated tooling sensors is a National Instruments (NI) CompactRIO, a reconfigurable real-time embedded control and data acquisition (DAQ) hardware. The chassis, shown in figure 3.2, consists of “Real-Time” and FPGA (Field Programmable Gate Array) levels of programming and can be configured with many different I/O modules for multiple control and monitoring applications, including for example general analog I/O, digital I/O, thermocouple input, and DC motor drivers [57], [60].

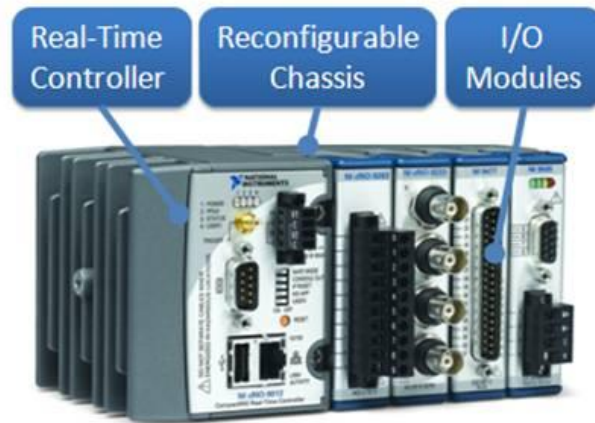


Figure 3.2 – NI CompactRIO chassis with reconfigurable FPGA and Real-Time capabilities and removable I/O modules. [57]

The specific CompactRIO (cRIO) that is used in this system is a NI cRIO-9022 chassis with a 533 MHz processor, 2 GB of hard drive storage and 256 MB of RAM storage. The chassis is quipped with dual Ethernet ports, a high-speed USB port, and a RS-232 serial port, which can all be used individually to connect to a host PC. It has a dual 9-35 VDC supply input for additional power [61]. The power to the cRIO-9022 is supplied through an AC/DC adapter that can be plugged into any normal electrical outlet.

The CompactRIO has been used in a wide variety of applications for sensor monitoring and feedback control. For example, this specific equipment was used in combination with this force sensor to perform model-based prediction and control experiments on the deflection of slender bars in the turning process [46] and adaptive control of cutting force to update force models using feedback from the force sensor [47]. Also, power monitoring was performed using the cRIO to update a machining power model using feedback from the force sensor [49].

The I/O modules that were used with this cRIO chassis were a NI 9215 Analog Input (AI) module and a NI 9211 Thermocouple Input module. The NI 9215 AI module has four 16-bit ± 10 V differential channels, either with screw terminals or BNC connection terminals, which can be sampled at 100 kilo-samples per second per channel [62]. The NI 9215 module that was used in this system allowed for screw terminal connections. The NI 9211 Thermocouple Input module has four 24-bit ± 80 mV differential channels specifically designed to read raw voltage signals from thermocouples of any type [63]. The actual cRIO that was used in the experimental setup can be seen in figure 3.3, and the individual I/O modules that were used can be seen in figures 3.4 and 3.5.

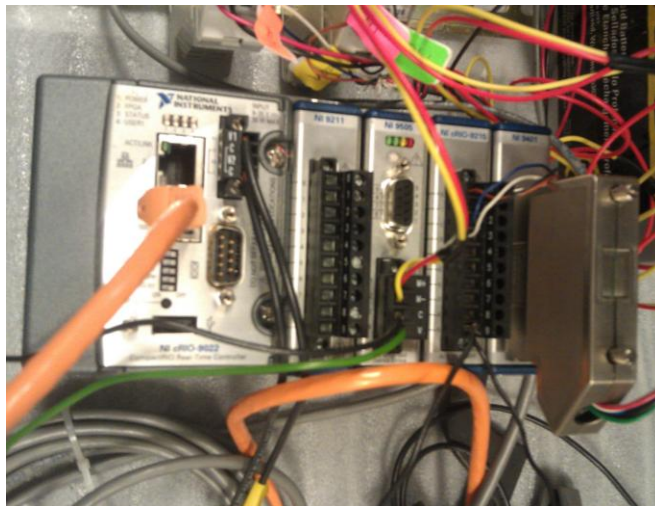


Figure 3.3 – NI cRIO-9022 chassis with several different I/O modules in place.



Figure 3.4 - NI 9215 Analog Input module with either BNC connection terminals (left) or screw terminal connections (right). [62]



Figure 3.5 – NI 9211 Thermocouple Input module. [63]

The cRIO can be used on Real-Time or FPGA levels, depending on which configuration is selected through the initial setup in the data acquisition software, LabVIEW. If the Real-Time level is selected for programming, the cRIO chassis will be configured to operate in “Scan Mode” which will search for any I/O modules that can be

read through this mode. The maximum sampling rate for the Real-Time level is 1 kHz. If the FPGA level is selected for programming, a much higher maximum sampling rate can be reached (up to 400 MHz for single-operation applications). The FPGA chip is re-programmable for custom applications, but can take some time to be compiled. For applications which do not require high sampling rates, programming on the Real-Time level is more beneficial than programming on the FPGA level due to the longer compile time. However, if an application does require a higher sampling rate than 1 kHz, the FPGA level would be the better option for collecting the raw data samples. A combination of the FPGA level and Real-Time level is optimal for fast data collection and short compile time because the raw signals could be collected on the FPGA level and the more complex computations required could be performed on the Real-Time level to save on compile time. However, the data collected on the FPGA level is buffered on the Real-Time level for analysis which can be a concern when acquiring accurate measurements is desired. A software loop time, which waits a specified amount of time (usually milliseconds) before continuing the next iteration, can be set in addition to the sampling rate if buffering is desired for slower data updating for monitoring live in the application.

Data Acquisition Software

The data acquisition software that accompanies the NI cRIO is NI LabVIEW. The version of LabVIEW that is used in this system is 2010 SP1 along with the Real-Time Module and FPGA Module, which are software toolkits that are required if the software is to interact with hardware that uses each of these levels of programming. LabVIEW is a

graphical programming development environment that can be used for standalone simulations and analysis, or integrated with NI DAQ hardware such as the cRIO to perform data acquisition for monitoring and controls applications [64]. In LabVIEW, the user will build a project with access to the various programming target levels (the host PC, Real-Time target, and FPGA target) in which to develop Virtual Instruments (VI), the LabVIEW programming environment. Each VI has two components: the Front Panel and the Block Diagram. The Front Panel is where all of the controls, indicators, graphs, etc. are located and serves as the user-interface. The Block Diagram is where all of the background programming logic is located, consisting of function blocks connected with “wires” to form the logic flow of the code. These two components work together to provide a unique, graphical interface with powerful programming capabilities.

For this system, three separate VIs were developed on each level (host PC, Real-Time target and FPGA target) which all work together to collect, analyze, and transmit data. The Project Explorer, figure 3.6, gathers all of the VIs and other dependent code for each level along with each target and its components and organizes it under a single project name and location. The user can adjust the properties of each target, view which modules are plugged into the target, and adjust the properties of each I/O port on each module as well.

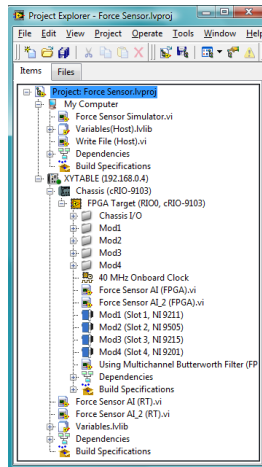


Figure 3.6 – Screenshot of the LabVIEW project explorer.

The FPGA VI, as shown in figure 3.7, acquires the signals from each I/O module at a specified sampling rate. The sampling rate can be controlled on the Real-Time (RT) level as an input to the FPGA VI. The RT VI, as shown in figure 3.8, reads the signals from the FPGA level and performs all necessary calculations on the data, which are then stored in *shared variables*, also known as *network variables* or *shared network variables*, over the NI Shared Variable Engine. This is a NI-built server to host shared variables over a network to be accessed by other LabVIEW VIs or NI products such as NI Measurement Studio, which was used in the development of this custom software application. The NI Shared Variable Engine can be accessed via the NI Distributed System Manager, figure 3.9, which is a portal to the server to be able to organize and view each shared variable on each target, usually the host PC's network, termed *localhost*, or the Real-Time target. The Host VI on the host PC, shown in figure 3.10, reads the data from the shared variables and allows the user to save the data if desired.

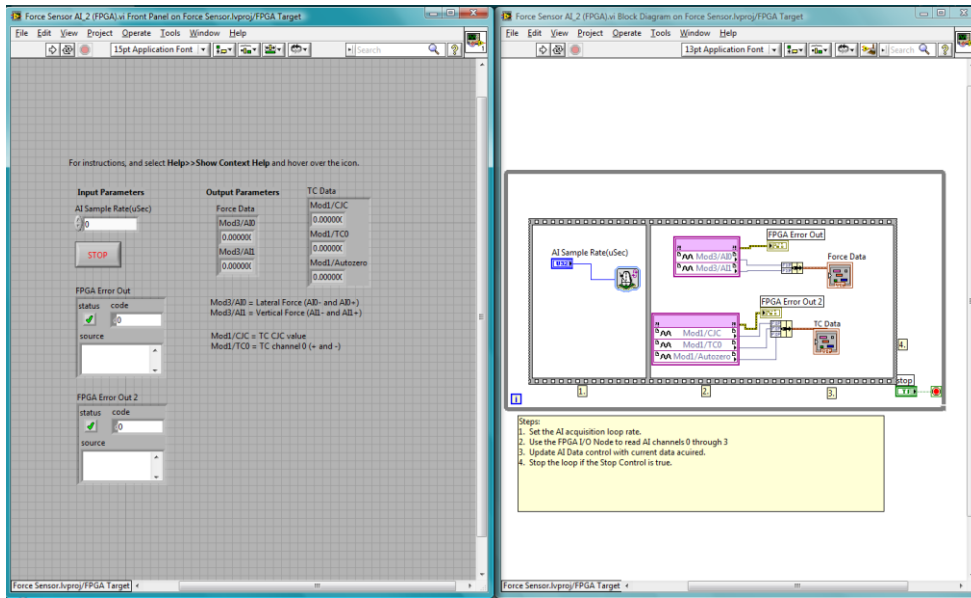


Figure 3.7 – Screenshot of the Front Panel (left) and Block Diagram (right) components of the VI on the FPGA level.

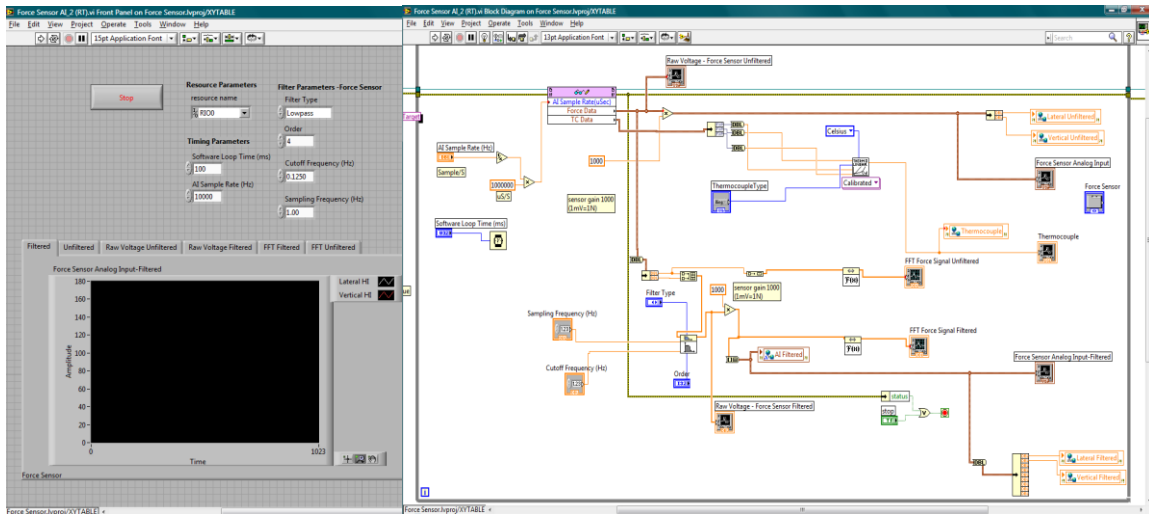


Figure 3.8 – Screenshots of the Front Panel (left) and Block Diagram (right) components of the VI on the Real-Time level.

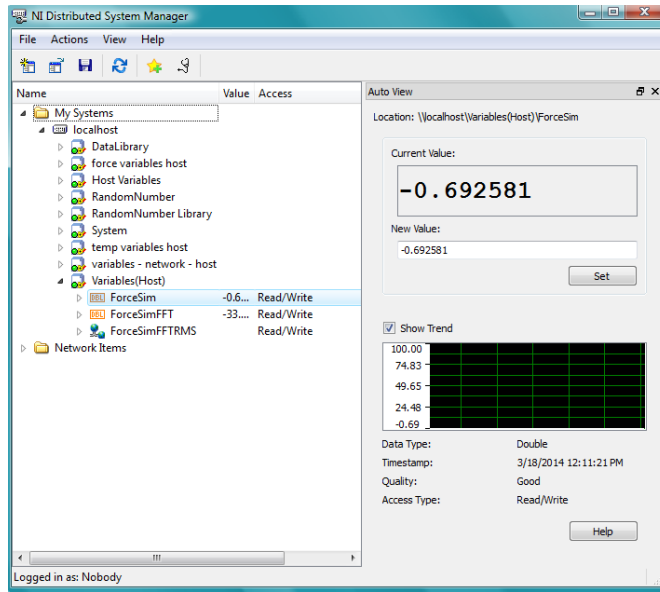


Figure 3.9 – NI Distributed System Manager shows the values of each shared variable.

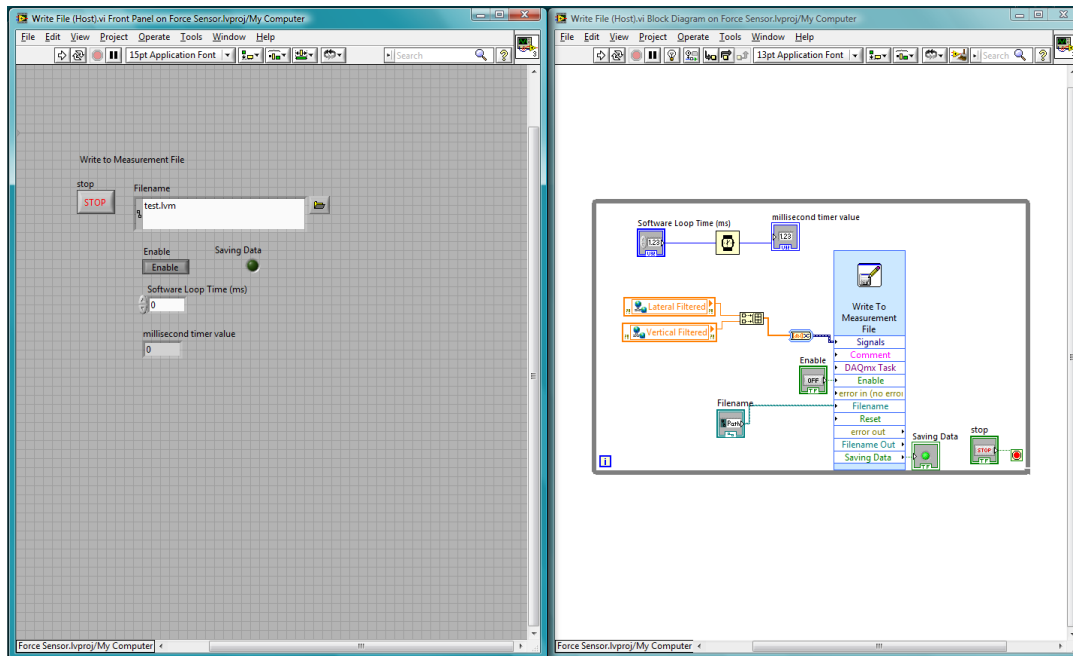


Figure 3.10 – Screenshot of the Front Panel (left) and Block Diagram (right) components of the VI on the host PC level.

The LabVIEW installation comes with several other software components that were useful in developing this specific system. One of those components is NI-MAX (Measurement and Automation Explorer), figure 3.11, which enables the user to view connections and the software installed on various hardware units. In order to interface with the cRIO hardware, the NI-RIO driver is also required to be installed on the host PC and the cRIO itself.

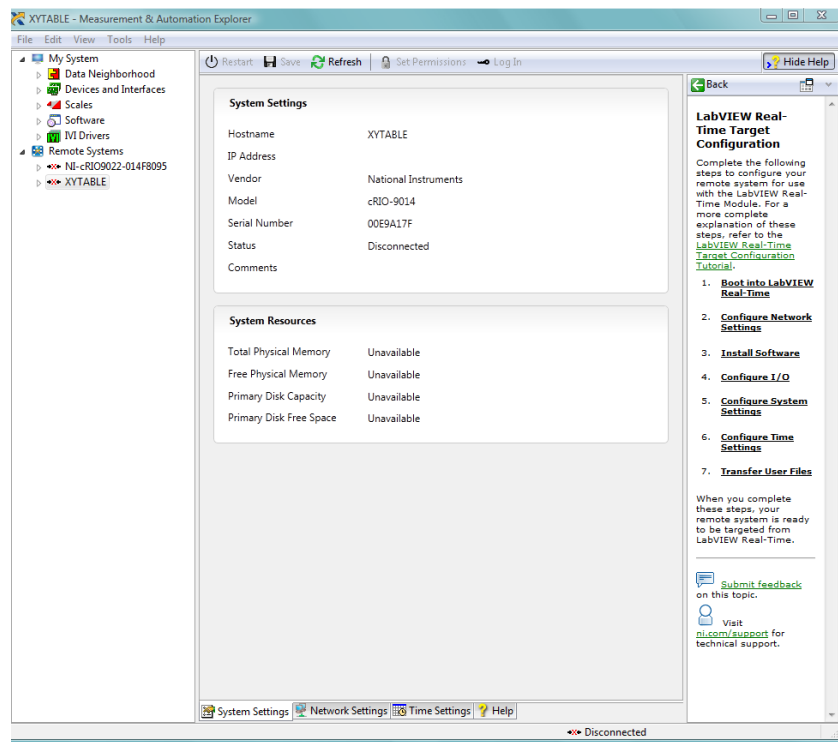


Figure 3.11 – Screenshot of the NI-MAX tool which shows the connections to various NI hardware.

MTConnect Installation and Setup

Okuma developed an MTConnect adapter and agent software kit for their customers to use on specific OSP controllers. The “Okuma MTConnect Adapter” interfaces with the THINC Application Programming Interface (API) on the machine’s controller to gather information about the machine’s parameters from the proprietary format in the controller. The adapter software includes a Graphical User Interface (GUI), seen in figure 3.12, which allows the user to view the components of the machine when they are gathered from the THINC API and view the connection to the agent when available. Once the adapter is running on the machine’s PC, which can be set to run at start-up, the user must run the agent executable through the command window (CMD). This window must stay open throughout the entire MTConnect connection so that the agent runs consistently. The agent sends a PING to the adapter, which is a message signal that is used to test the communication between two entities of a network connection. Once the adapter receives the PING from the agent, it sends a PONG back to the agent. When the agent receives the PONG, it sends another PING to the adapter. This process, which occurs in every 1 second, makes sure that the connection is still present and displays an error message in the adapter GUI when this connection has failed.

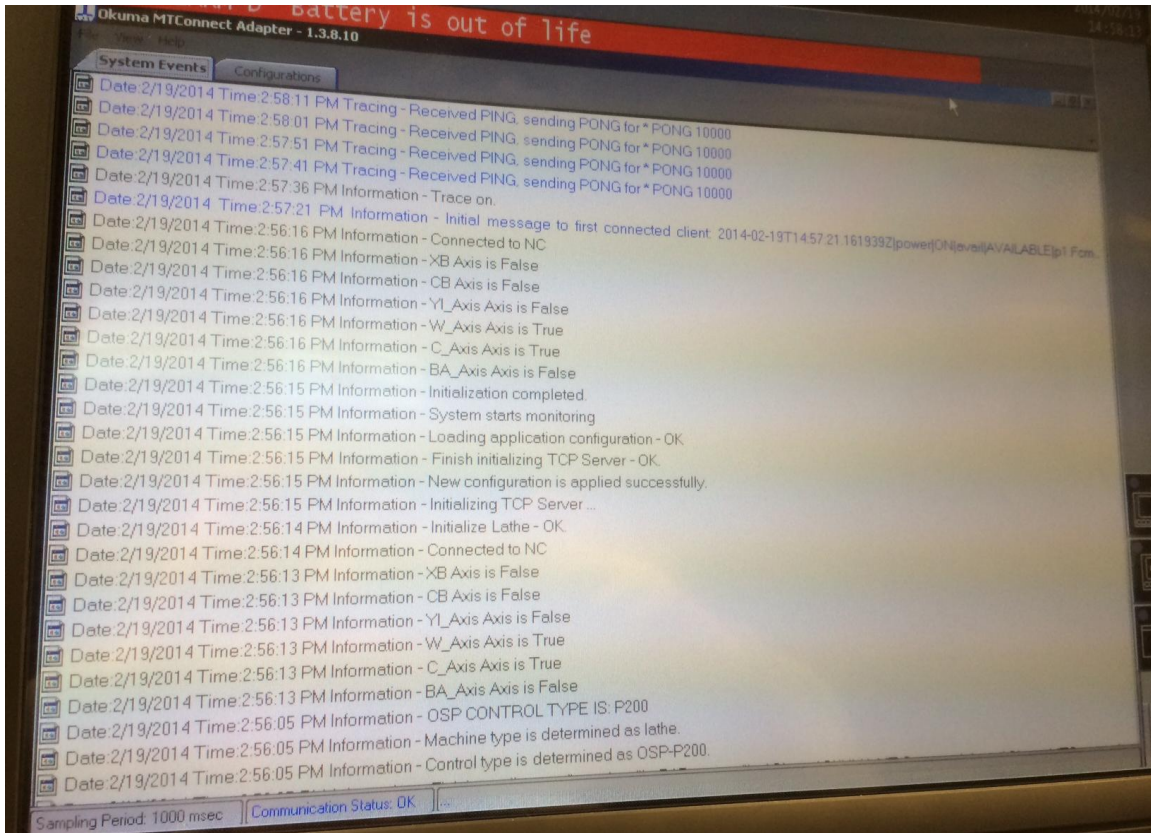


Figure 3.12 – Screenshot of MTConnect adapter GUI on the machine PC.

In order for any application to access the agent, which will provide the relevant information from the machine in the standard MTConnect format, the machine PC must be connected to an Internet or Intranet (internal network) connection. In this system, the machine PC was connected wirelessly to the university network. This allows an application running on the same network to connect to the agent. If it were desired to connect to the agent over any Internet connection from any host PC, the machine PC would have to act as a server to host the agent and its associated data that would then be streamed over the Internet. This requires extensive hard drive space, which this machine

PC does not have, or an additional PC that would host the server independently, which was not available. Creating a custom server also requires complex programming abilities. It also exposes the machine to threats from viruses, hackers, and other security issues that would need to be blocked with extensive anti-virus capabilities. If this were desired by a manufacturing facility, they could implement their own server to have the data accessible through any Internet connection, or they could host their agent(s) on their internal facility's network to be accessed only at the facility.

For this particular system since a custom server was not available to host the agent data streams over the Internet, both the agent and the host PC must be connected to the same university network to allow the application on the host PC to access the machine information through the agent. The agent's address is simply a URI which takes the form of a typical web address. In this particular system, the agent address requires the knowledge of the IP address of the machine PC. An example of the agent address is "http://123.45.678.900:5000/OKUMA.Lathe" where the IP address is "123.45.678.900" on port 5000 connecting to OKUMA.Lathe. The IP address of the machine in this system was a dynamic IP which changed each time the machine was powered on. A static IP address could also be set up on the machine if desired.

Since the agent's address takes the form of a web address, any PC connected to the same network as the agent (if it is not hosted on the Internet through a server) can simply enter the address into a web browser to see the agent XML structure. This also means that any application can connect to the agent in the same way and gather the

information from the agent structure to perform functions on the data and display it in any way that is desired.

There are many methods for connecting to the agent and using the information it contains, and many types of software development tools that enable a developer to create applications to access this information. This is open to the developers themselves to come up with creative applications to utilize the information through MTConnect. The next chapter describes the development and architecture of an MTConnect software application.

CHAPTER FOUR

SOFTWARE APPLICATION DEVELOPMENT & ARCHITECTURE

This chapter will discuss the process that was involved in developing the software application involved in this system. This process began with identifying suitable tools to develop the application and its associated user interface to monitor the machining process in the way defined by the project statement and objectives. In addition, several function structures were created that define the architecture and logic flow of the application in order to guide the software development process and provide some insight into the functionality of the application.

Software Development Tools

The software development process typically involves working in an Integrated Development Environment (IDE), which is a software package that allows a developer access to a text editor, compiler, and debugging tools to develop the underlying code for the application. In visual programming, which allows the developer to manipulate application elements in a graphical manner instead of writing the code in a text editor, the IDE may also have features to assist in designing the appearance of a user-interface.

In order to develop a software application with a functional, user-friendly, and easy-to-operate interface, several IDE environments were considered based on a few criteria. Since the data collected from the MTConnect agent is stored in an XML structure, the IDE would have to allow the data to be parsed from the XML structure, i.e. the data is converted into a specific data type that can then be manipulated in the desired

programming language or environment. Also, the application would need to be able to monitor the signals from the sensors which would be read external to MTConnect, so the application would need to connect to the CompactRIO to read this data. Lastly, if the application were to be used outside of the IDE, it would require the deployment of an executable file with the extension “.exe” which would allow the application to run independently of the IDE. This is necessary if a standalone application is desired with the ability to distribute the application to other users.

Based on these criteria, a few IDE options were considered, as seen in figure 4.1. If NI LabVIEW was used on its own, it could have the ability to parse the XML structure from the MTConnect agent and monitor the external sensor data with a direct connection to the CompactRIO. However, it does not have the ability to deploy an executable without installing the NI Application Builder toolkit that requires an additional licensing fee which can be expensive.

	NI LabVIEW ONLY	Visual Studio ONLY	Visual Studio w/ NI Measurement Studio
Parse XML from Agent?	✓	✓	✓
Monitor external sensor data?	✓	✗	✓
Deploy .exe?	✗	✓	✓

Figure 4.1 – Several IDE options considered to develop the application based on certain criteria that were necessary for its proper functionality

Another IDE that was considered was Microsoft Visual Studio, which allows developers to code in several popular programming languages including Visual Basic (VB.NET), Visual C#, and Visual C++. Since Visual Studio is a Microsoft product, it allows developers to create applications in the familiar Windows interface. This IDE was considered as an option after reviewing a tutorial, “Building a Client” [65], provided by the MTConnect Institute on the MTConnect website [3] with the ability to download the source code from CodePlex [66], a website for downloading open-source software code from various users. This tutorial was developed as a workshop for one of the the annual MTConnect “MC2” conferences. The client, or software application that connects to MTConnect, that could be developed from the tutorial was meant for people interested in MTConnect to use as a starting point for developing their own applications for their individual needs. This basic application, referred further as the “MTConnect Sample Client”, included the ability to connect to any agent and parse the XML to be used in several ways including displaying the data in lists, grids, and on a graph. However, since the Visual Studio IDE is a Microsoft product and is not affiliated with National Instruments directly, using Visual Studio alone to develop this application to monitor machining process information from MTConnect as well as external sensor data was not an option.

After researching additional IDE options that would meet all of the criteria for developing this application, a solution was found with NI Measurement Studio for Visual Studio. NI Measurement Studio is a toolkit with libraries of code and elements of LabVIEW’s graphical controls and indicators that integrate into the Visual Studio

development environment. This would allow the development of an application that could parse the XML structure to collect data from the agent, deploy a standalone executable, and connect to the CompactRIO through the NI toolkits and code libraries. Therefore, the final choice was made to use Microsoft Visual Studio with NI Measurement Studio to be able to meet all of the desired criteria.

Several programming languages can be chosen for application development in Visual Studio. The MTConnect Sample Client [65] was developed using Visual C# so it was natural to continue in this programming language for convenience. The Visual Studio IDE interface contains a text editor which can be viewed in a tabbed window in the center of the screen, as shown in figure 4.2, along with a Solution Explorer that shows the structure of the entire project and its associated files and solutions. When a project is compiled, a solution file is created which contains all of the code dependencies. Multiple projects can be developed under one solution file and use the same dependencies. Also, multiple “Windows forms” can be used to separate the code into pieces for better organization and easier debugging for the developer. In addition to viewing and editing the underlying code, the developer can view the “Windows form”, shown in figure 4.3, to design the appearance and functionality of the user-interface. The “Windows form” is familiar to any Windows user with the same type of appearance, buttons, text boxes, lists, and other elements that are used in common Windows software applications. Also, developers can click and drag elements from a toolbox and place them anywhere on the form. These elements can be resized and designed with any type of font or color for the

desired appearance. Other properties can be altered as well in the Properties window which changes depending on which element is currently selected.

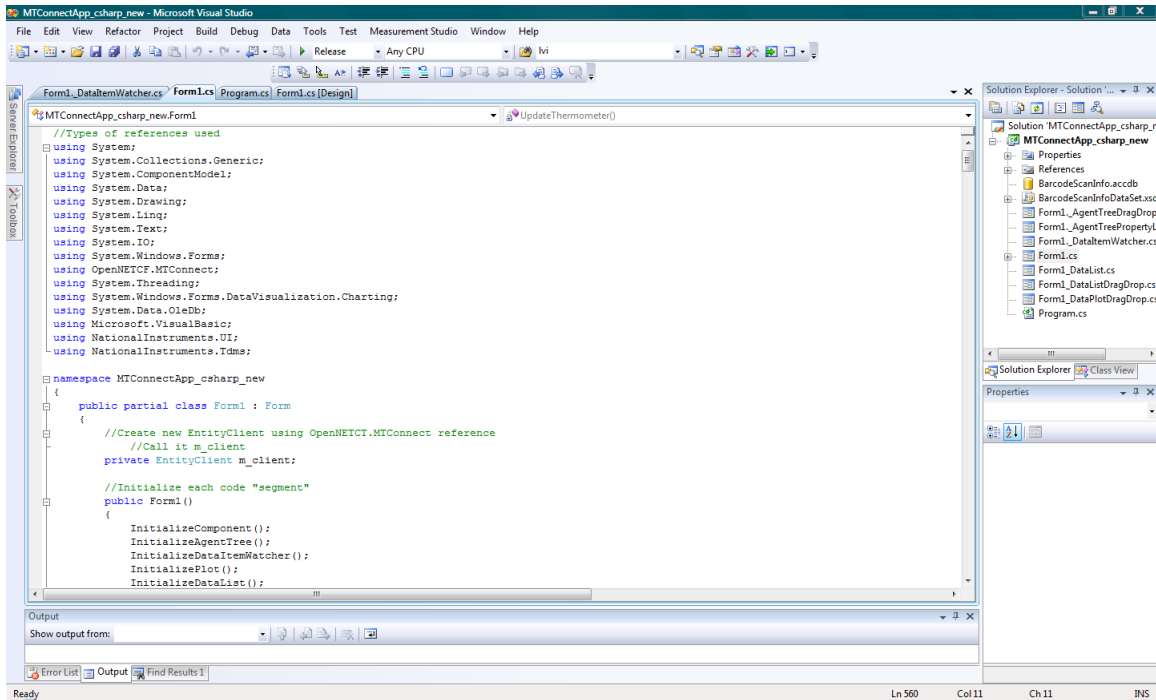


Figure 4.2 – Screenshot of the Visual Studio 2008 IDE showing the C# code development tab (center) and Solution Explorer (top right).

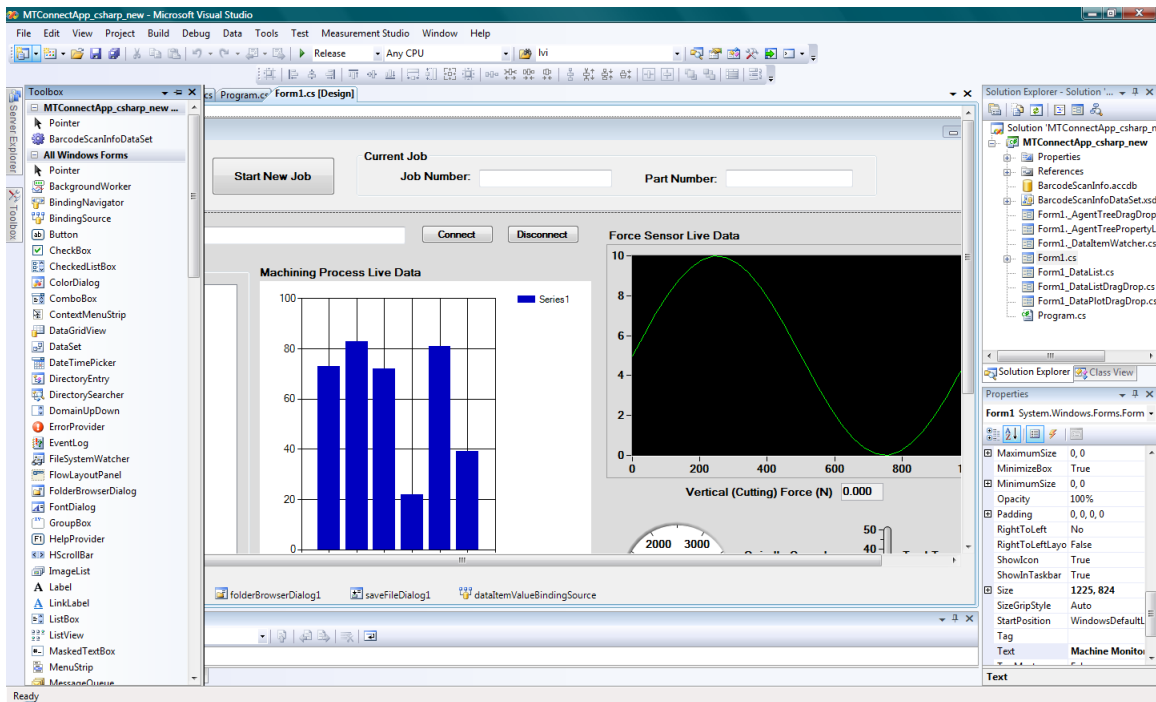


Figure 4.3 – Screenshot of the Visual Studio 2008 IDE showing the “Windows form” design (center), Properties box (bottom right), and Toolbox (left).

Once the desired elements are placed on the Windows form, the developer can write the code necessary to achieve whatever functions are desired. Because C# is an object-oriented programming language, Visual C# treats these elements as objects that can respond to certain events such as the click of a button, text change, the enter key being pressed, etc. This allows the user to interact with the application directly, such as clicking a button to perform a certain function or typing text into a text box. This ability allowed the software application to be developed with the user in mind, making it simple to use and intuitive which would help increase operational productivity in the machining process.

Application Functions and Architectures

The development process detailed above was used to create two distinct applications related to improving manufacturing operation quality. The first, at request of GE Power and Water in Greenville, SC, was a “Work Setup” application for data input accuracy improvement. The second was for detection of potentially damaging dynamic machine conditions using the MTConnect standard. This application built upon the MTConnect Sample Client from the “Building a Client” tutorial provided by the MTConnect Institute [65], used Measurement Studio to incorporate the integrated tooling sensor data, and used elements of the “Work Setup” application that was developed for GE. The architecture and logic flow of these distinct applications will be discussed in this section.

Work Setup Application for GE Power & Water

After learning about the development of this application for monitoring and communicating machining process information, a partner in Manufacturing Technology at GE Power & Water expressed a need for part-tracking capabilities in their machine tools for manufacturing gas turbine blades out of high-value materials in their Greenville, SC facility. They requested a barcode scanning application that would help them track each job operation performed on specific parts by individual personnel. A separate application was developed specifically for their use, but elements of it were also used as a framework for the final application. They have also expressed a desire to integrate

MTConnect on their machine tools and are interested in customizing both of these applications for use in their manufacturing facility in Greenville.

The separate application developed for GE was based on their requirements list with the following criteria. The application would prompt the operator to scan their badge with a barcode scanner, an Intermec brand SR61T model tethered industrial handheld scanner [68], shown in figure 4.4. Next, the application would prompt the operator to scan the Job Number barcode and then the Part Number barcode. Based on the part number, the application would load the appropriate part file and work instructions to assist the operator in performing the required machining operation according to that part's specifications. The application would automatically save this information (badge number, job number, and part number) in the machine controller's *Common Variables* as well as in a database to track which operator performed certain jobs on specific parts. This would ultimately help them organize their operational procedures and aid in their post-process analysis of specific machining operations.



Figure 4.4 – Intermec SR61T barcode scanner for the GE Work Setup application [68].

A function structure, shown in figure 4.5, was created to demonstrate how the application should function and what the specific tasks were to be performed. This helped in creating the logic and developing the underlying code for the application, which was also developed in Microsoft Visual Studio. At first, this code was developed using Visual Basic (VB.NET) after reviewing tutorials on how to develop applications in VB.NET that interface with the Okuma THINC API and allow users to save data to the Common Variables on an OSP-P200 controller [69], [70]. However, the elements of this application that were used in the final MTConnect application were converted to Visual C# so that the basis of the MTConnect Sample Client could be used.

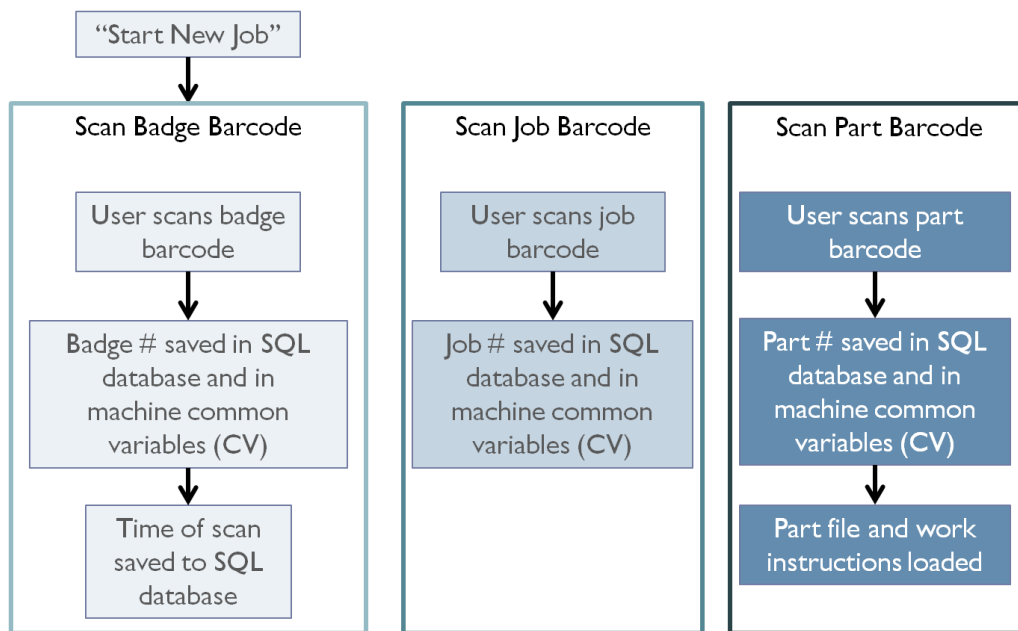


Figure 4.5 – Architecture and logic flow of the Work Setup application requested by GE Power & Water.

When the “Barcode Scanner” application starts up, as shown in figure 4.6, the user will begin by clicking the “Start New Job” button. This will display a message box to prompt the user to scan their badge number, as seen in figure 4.7. The operator will scan their badge with the barcode scanner, which can be set up to “type” the barcode number in the text box. This is done by configuring the barcode scanner through its setup software to be used as a “keyboard input” which allows the scanner to act as a separate keyboard to “type” the barcode numbers in after its scans and reads the barcode. The scanner can also be set up through one of the PC’s COM ports, which is slightly more complex to program in the application.

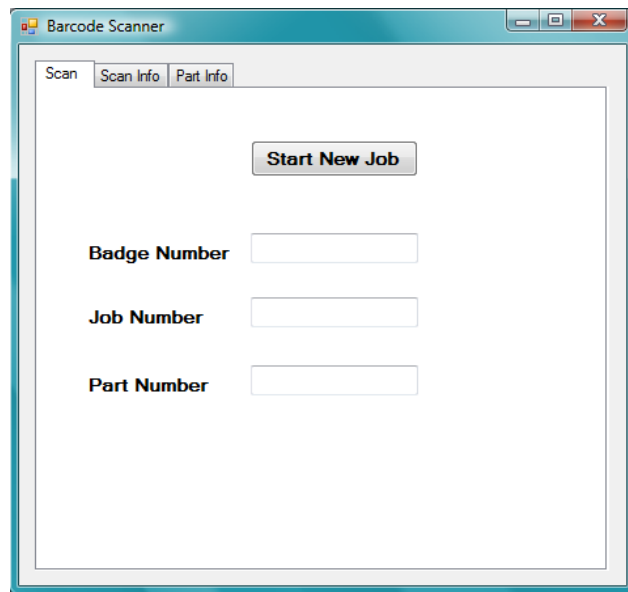


Figure 4.6 – Barcode scanner user-interface for the GE Work Setup application.

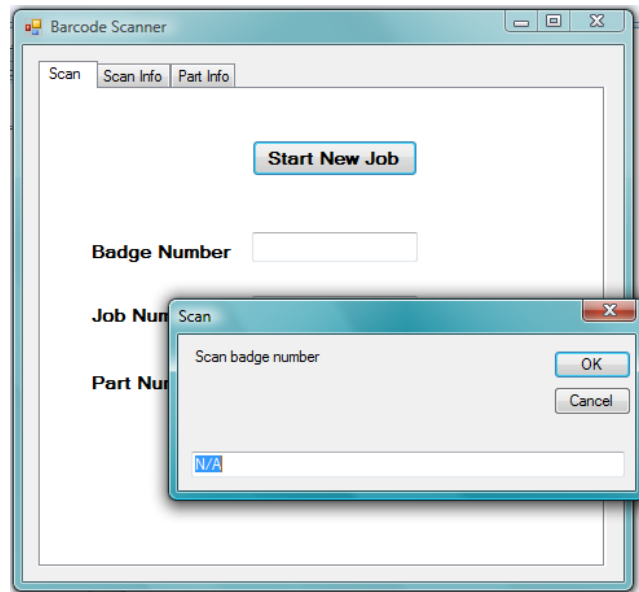


Figure 4.7 – Pop-up message to prompt the user to scan their badge.

After the badge barcode is scanned, the number appears in the “Badge Number” text box and is saved to the machine controller’s Common Variables and into the “Scan Info” database table, which can be seen by clicking on the “Scan Info” tab on the interface, as seen in figure 4.8. A Microsoft Access database was set up for use with this application, although any database software with Structured Query Language (SQL) capabilities can be used. SQL is a query language allowing data to be read from and saved to databases, among other functions [70]. Visual Studio can access SQL databases and perform these query functions in applications to display data from and save data to databases. The Access database software was used because it is compatible with Visual Studio and other Microsoft products. The application connects with the MS Access database that was set up, as seen in figures 4.10 and 4.11, and imports the tables into the application’s user-interface, seen in figures 4.8 and 4.9.

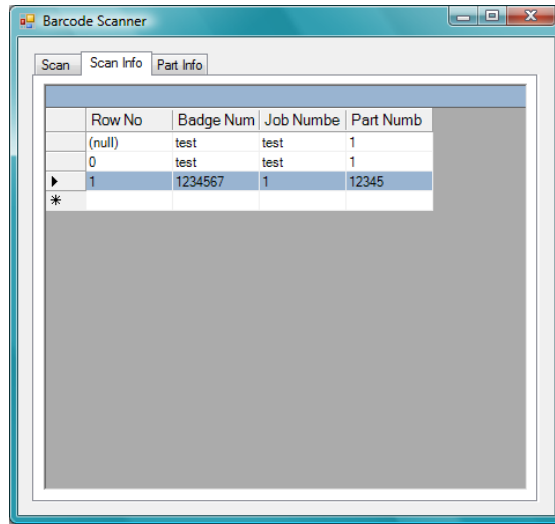


Figure 4.8 – “Scan Info” tab shows the database table that stores the badge number, job number, and part number after each scan.

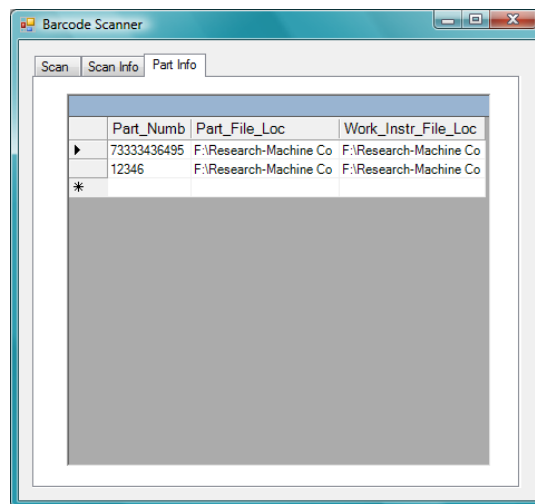


Figure 4.9 – “Part Info” tab shows the database table that stores the part file and work instruction file locations based on each part number. The application reads this table to search for a matching part number based on the part barcode that was scanned and opens the corresponding part file and work instruction file.

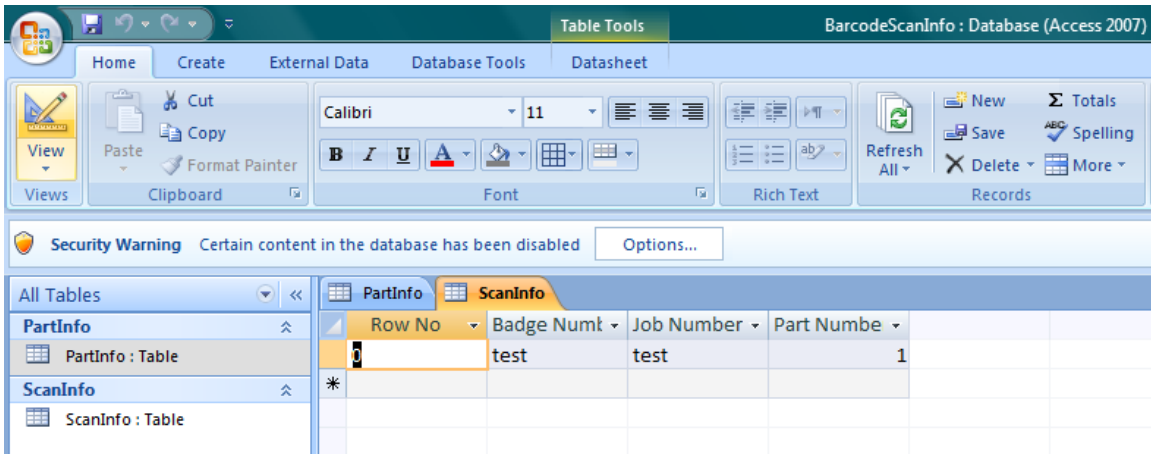


Figure 4.10 – “Scan Info” table in the MS Access database.

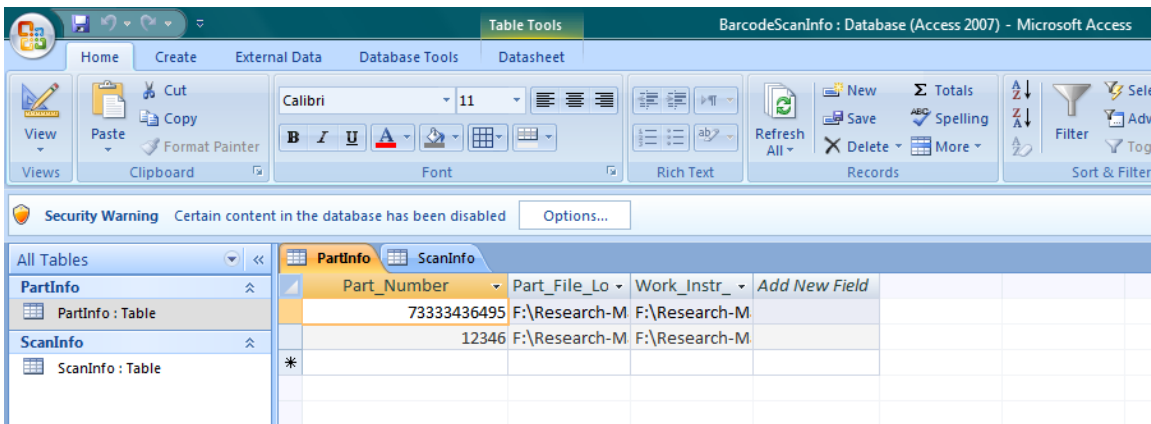


Figure 4.11 – “Part Info” table in the MS Access database.

After the badge number has been scanned, another message box will pop up prompting the user to scan the “Job Number” barcode, figure 4.12, and the application takes the same steps as before to save the job number in the “Scan Info” database table and machine’s Common Variables. The same procedure occurs for scanning the “Part Number” barcode, as shown in figure 4.13.

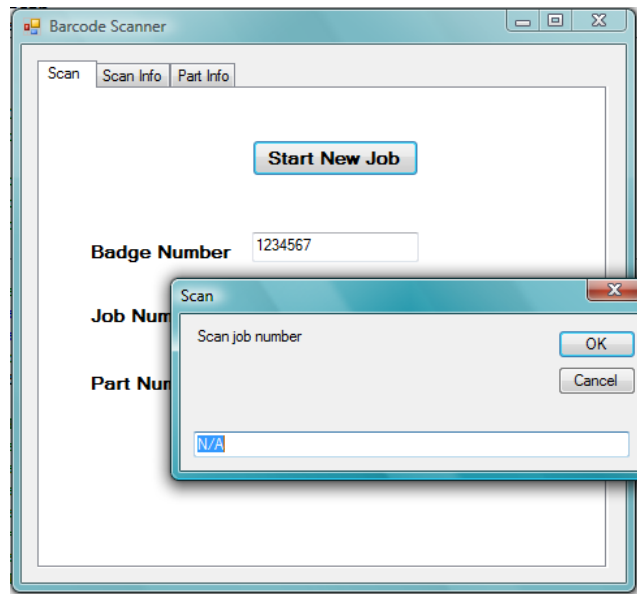


Figure 4.12 – Pop-up message to prompt the user to scan the job barcode.

Once the part number is entered into the “Part Number” text box, the application searches the “Part Info” database table, which can be seen by clicking the “Part Info” tab on the interface, as seen in figure 4.9. When the appropriate part number is found in the “Part Info” database table, the part file location and work instructions file location are read into the application which opens the files. For testing this application, sample PDF files were created and opened using the “Process.Start” function in VB.NET which interfaces with the user’s computer to start any process with a location or address, including opening files or web browsers.

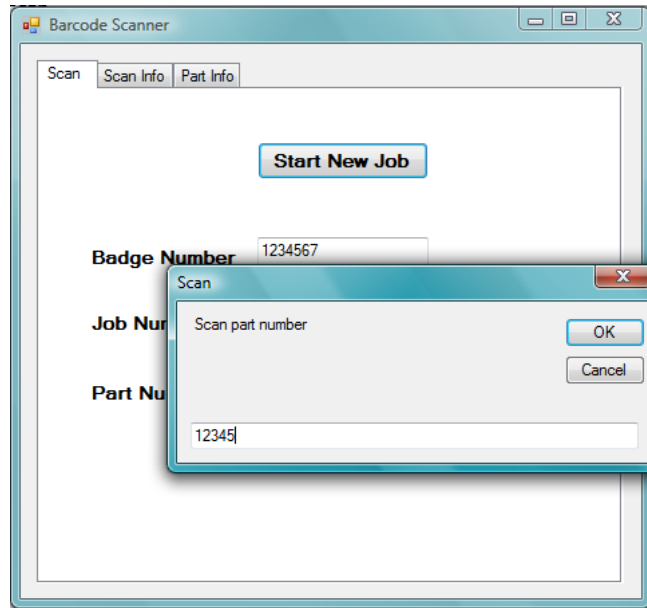


Figure 4.13 – Pop-up message to prompt the user to scan the part barcode.

This “Barcode Scanner” application was reviewed by the partner at GE and is now being tested on their machine tools and integrated with their internal SQL database at the facility. They are also interested in integrating MTConnect on their machine tools and customizing this application to include MTConnect capabilities, following the final application resulting from this project as a guideline.

MTConnect Sample Client

After reviewing the “Building a Client” tutorial from the MTConnect Institute [65], insight into the sample application was gained and a function structure describing the application architecture and logic flow was created. The basic functionality of the Sample Client was used as a starting point for this application, so it was important to

understand how the pieces of code worked together and determine what could be improved. Figure 4.14 shows the function structure of the MTConnect Sample Client on a high-level to present the details that are important in operating the application.

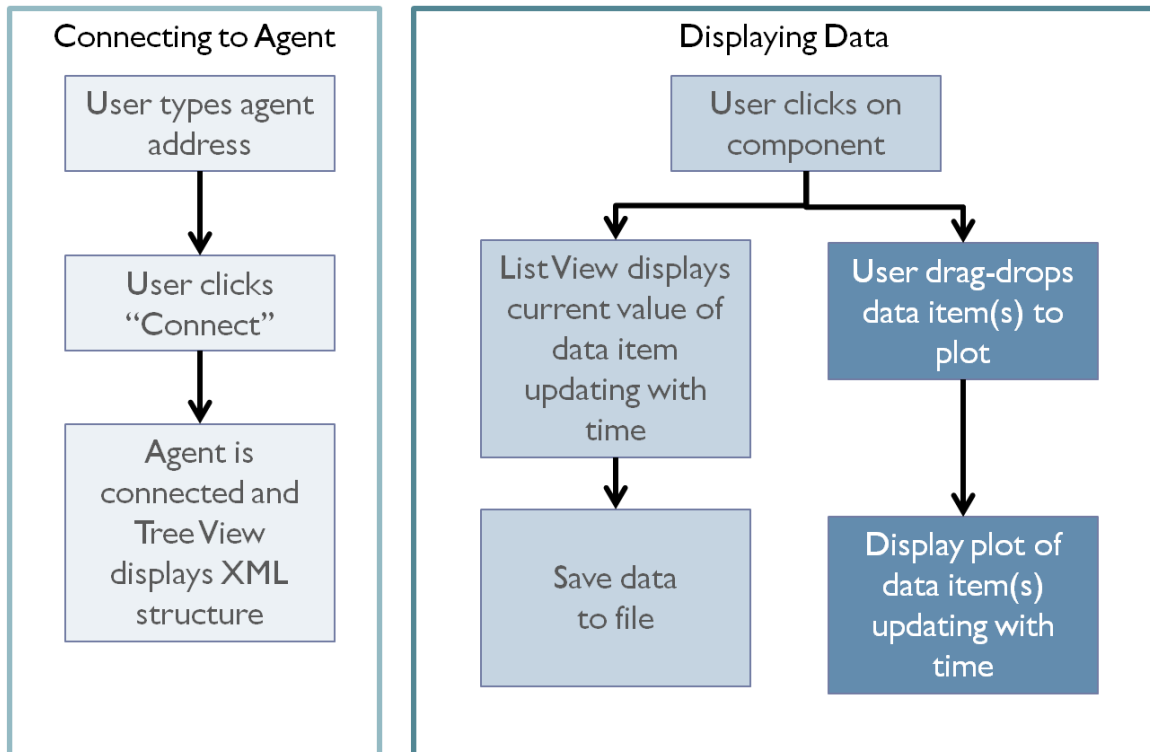


Figure 4.14 – Architecture and logic flow of MTConnect Sample Client.

The Sample Client application functions as follows according to figure 4.14. Once the application starts, the user types the address of the agent into the “Agent Address” text box and clicks the “Connect” button, as shown in figure 4.15. The button click event tells the application to connect to the agent address in the text box.

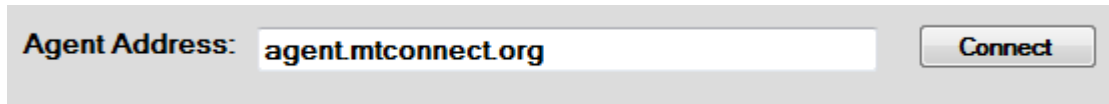


Figure 4.15 – “Agent Address” text box and “Connect” button. The user typed “agent.mtconnect.org” into the text box and would then click “Connect” to retrieve data from the agent.

Once a connection is made, the application sends a “Probe” command to the agent to gather the device’s collection of components and their associated data items. The code iterates through the entire agent structure to gather each component and each data item under each component, and the “Agent Structure” tree view is populated with this information in a hierarchy similar to the agent’s XML structure, as seen in figure 4.16. The user can click on any data item under any component in the “Agent Structure” and view its associated values and properties in a list view below the tree view. In addition, the user can “drag-and-drop” any data item into a second list view which will display its value and a few other properties. This list view, seen in figure 4.17, is meant for the user to gather any data items they wish to save in a file for future viewing and analysis.

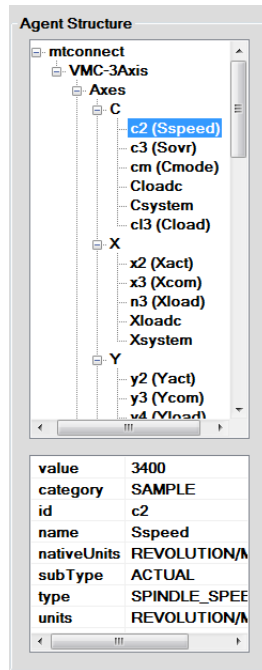


Figure 4.16 –The Agent Structure is displayed in a tree view (top) and the “c2 (Sspeed)” data item’s associated values and properties are displayed in the list view (bottom).

Category	ID	Name	Type	Value
<input checked="" type="checkbox"/> Sample	c2	Sspeed	SPINDLE_...	3400

Figure 4.17 – Data items can be dragged and dropped to this list view as well which will gather the information desired to be saved.

Also, the user can click and drag any data item into a plot, as seen in figure 4.18. The property list, data item list, and plot all update every 100 milliseconds according to the “DataItemWatcher” function in the code, which sends a “Sample” command to the agent to gather a stream of data item values.

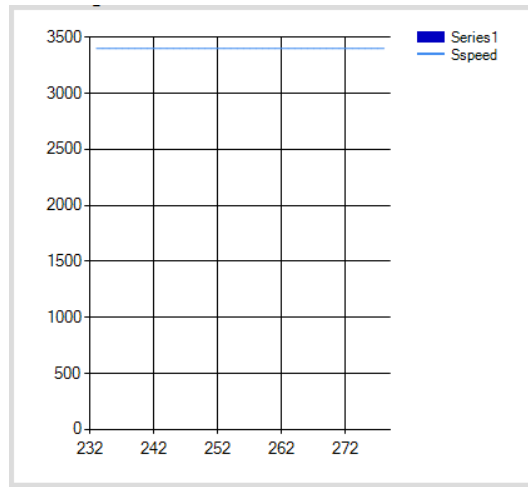


Figure 4.18 – The plot displays each data item that is dragged and dropped into it as a separate series.

Since this Sample Client was designed originally as a starting point for users to develop their own custom applications, the functionality is very basic. Once the user starts the application (either in debug mode or after building the executable), the user can only connect to the agent one time and cannot disconnect from the agent unless the application is closed out entirely. Also, once data items are dragged into the data list or the plot, the action cannot be undone unless the application is closed out entirely. This does not allow for the application to be totally functional if a user were to deploy it for multiple cutting operations, if they wanted to change the data items they recorded and plotted, or if they wanted to disconnect from the agent to either reconnect to the same agent or connect to a different agent. These additional elements were incorporated into the final application to make it easier to use and functional for multiple operations.

Chatter Identification Application Architecture

The final application is a combination of the MTConnect Sample Client, the fundamental part tracking abilities of the GE Work Setup barcode scanning application, and the addition of Measurement Studio controls to read the integrated tooling sensor data. All of these features allow the user to view all of the important data collected through MTConnect and through the external sensors in one central location. It also gives the user the ability to save all of this data in one file with the ability to track the part number and job number directly in the file name. This allows the user to organize the important part-specific information for future analysis and also gives the machine operator the ability to monitor the machining process live to correct for any part-damaging conditions that may arise.

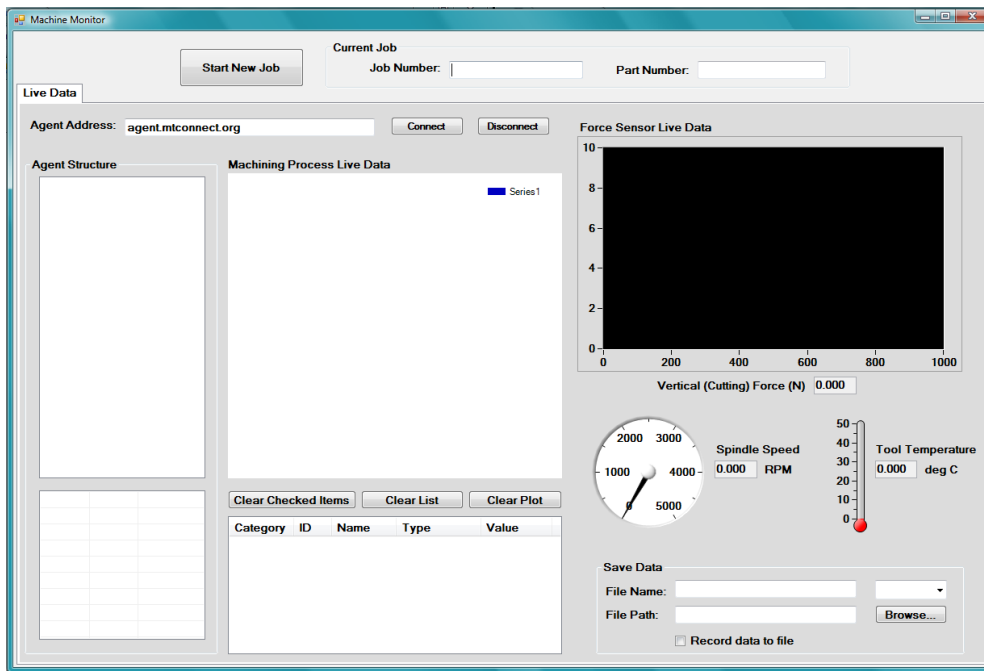


Figure 4.19 – Screenshot of the application's user-interface.

Figure 4.19 shows the user-interface of the application at start-up. The appearance of the application interface is similar to the MTCConnect Sample Client, except for the additional buttons and controls that help extend the functionality of the original Sample Client and include the part tracking abilities of the GE Work Setup application, and the Measurement Studio controls that help integrate the sensor data from the CompactRIO.

A function structure was created for the final application to organize the logic flow and show the architecture at a high level. Figure 4.20 was created based on the combination of the function structure of the MTCConnect Sample Client, figure 4.14, and the function structure of the GE Work Setup application, figure 4.5 based on what was required for this final application to meet the project objectives. This helps to organize the architecture into categories to assist in describing each major function of the application.

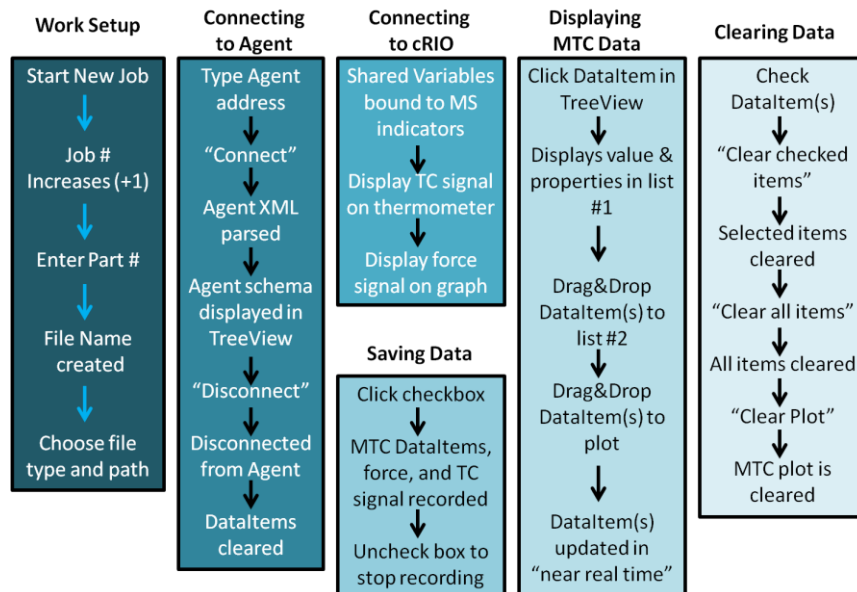


Figure 4.20 – Overall application architecture separated into its major functions and sub-functions.

Each block in figure 4.10 shows a major functional area of the application with its associated sub-functions. The “Work Setup” function is the process to start each job and set up the application to record and save data based on a specific part number and job number. This includes clicking a button to start a new job, scanning the part barcode when prompted, and choosing a file type and location to save the file based on the name that has been created automatically in the application based on the part and job numbers. The “Connecting to Agent” function consists of typing in the agent address and clicking a button to connect to the agent that will parse the agent XML structure to display data in a tree view in the application, in the same way that the MTConnect Sample Client displayed this information. Once the agent is connected, the user can disconnect from the agent if desired which will clear all of the data items from the tree view, both list views, and plot that could contain MTConnect data.

The “Connecting to cRIO function” involves the connection and display of the data from the CompactRIO and background LabVIEW programs that read the integrated tooling force sensor and thermocouple. The associated shared variables that host the signals from the sensors are automatically bound into the application’s Measurement Studio indicators when the application begins running. As long as the shared variables are collecting the signals from LabVIEW, the data will be displayed in the application.

The “Display MTC Data” function consists of clicking a desired MTConnect data item in the agent tree view which will display the item’s properties and value in a list view (list view #1 according to figure 4.20). The user can then click and drag the data item to a second list view (list view #2) or a plot, both which will update the data item’s

value in near-real time with the same DataItemWatcher as in the MTConnect Sample Client.

The “Saving Data” function contains the sub-functions that allow the desired data to be saved to a file. The user would click a checkbox which would initiate the recording of the data items that were dragged to list view #2 alongside the force sensor signal and the tool temperature data from the thermocouple signal. When the user wants to stop recording data, they would click that same checkbox again to “uncheck” it. The “Clearing Data” function allows the user to clear selected data items from list view #2, clear the entire list, or clear the plot of all MTConnect data items.

Details of the operation of the major functions of the application are discussed further in Chapter 5, along with an overview of the entire system’s operational procedure.

CHAPTER FIVE

SYSTEM OPERATIONAL PROCEDURE FOR THE CHATTER IDENTIFICATION APPLICATION

This chapter will discuss the data communication process of the system used for the chatter detection application with external hardware and software integrated to the machine tool using the MTConnect standard, as well as the system's full operational procedure, including the operation of the software application's user-interface.

System Data Communication Process

The technical capabilities of this application can be described while describing the full system operational procedure, which is somewhat specific to the machining setup in our laboratory. A schematic of the application's data communication flow can be seen in figure 5.1. The machine tool (Okuma Horizontal CNC Lathe) with its PC-based numerical controller (THINC OSP-P200L) contains the MTConnect Adapter and Agent software components. As mentioned in chapter 3, the adapter and agent communicate PINGs and PONGs in order to establish a connection and transport data to and from the machine in the standard MTConnect format. The application, created through Visual Studio, on the Host PC communicates with the MTConnect agent with the "Probe" and "Sample" commands to gather the machine tool data items that are supported by the MTConnect standard, such as the spindle speed, feed rate, tool position, and other parameters. This transmission of data is over the university's internal network using the machine's IP address to connect to the agent.

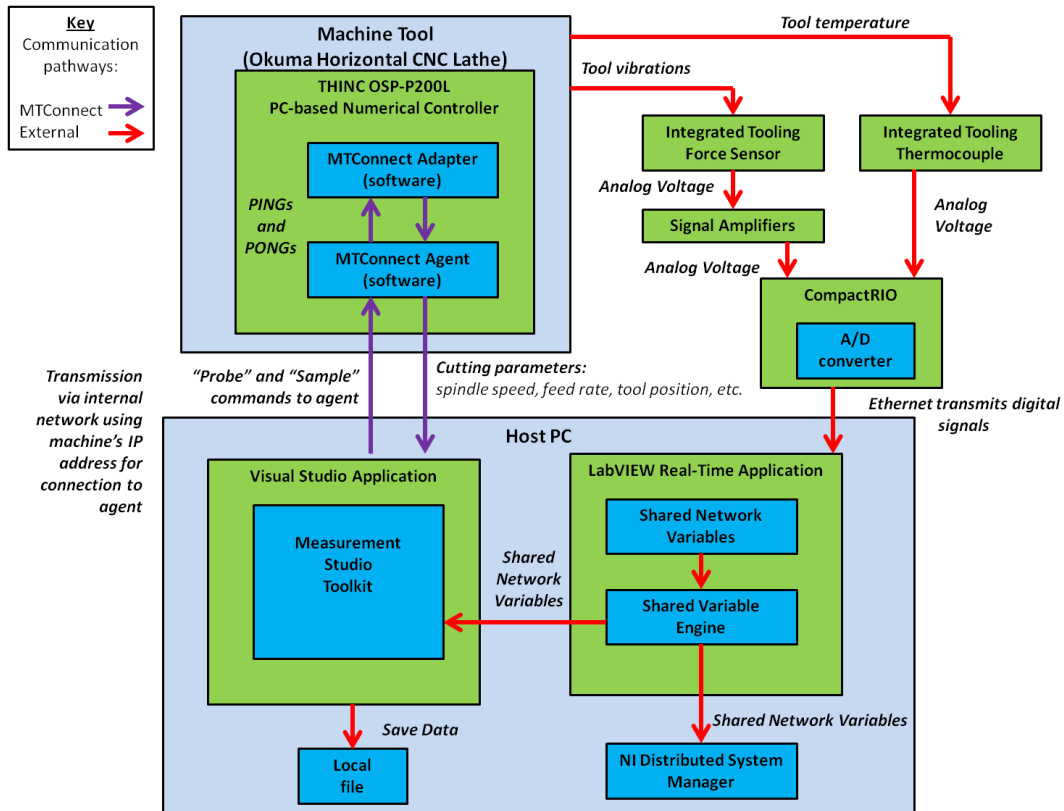


Figure 5.1 – Schematic of the data communication flow between various components of the system.

The integrated tooling sensors, including the force sensor and thermocouple, sense the tool vibrations and temperature. The force sensor converts the vibrations to analog voltage through its strain gauges which are sent through the signal amplifier to the CompactRIO. The thermocouple senses a change in temperature and outputs a change in analog voltage which is also sent to the CompactRIO. These analog signals are converted to digital signals in the CompactRIO on the FPGA level, processed on the Real-Time level, and sent to the Host PC over an Ethernet connection. The LabVIEW programs on the Host PC, consisting of the FPGA, RT, and Host VIs, acquire the signals from the

sensors and perform the necessary calculations. The data is sent to appropriate “Shared Network Variables” and hosted on the NI Shared Variable engine. These shared variables can be viewed in the NI Distributed System Manager and are sent to the Measurement Studio indicators in the application created in Visual Studio. The data that is collected through MTConnect or the external sensors can be saved through the application to a file location existing on the Host PC.

Operational Procedure

The user must first turn on the machine tool (Okuma LB4000EX) and wait for the machine’s numerical controller (the OSP-P200L NC) to load. After the OSP NC is loaded, the Okuma MTConnect Adapter should start up at this time. If it does not start automatically, the user can do this manually via the start menu and going into the programs on the PC. The user must also identify the machine’s IP address which is likely to change each time the machine is turned on if it is not set as a static IP address. The user can access the IP address using the command window and the command “ipconfig”. Also in the command window, the user must navigate to the folder in which the adapter is located in order to run the agent. Once in the correct directory, the user must run “agent.exe debug” to start the agent. By checking the adapter GUI, the user can make sure that the agent and adapter are communicating the appropriate PINGs and PONGs every second. This is the setup for the MTConnect capabilities of the application, which is simple and only necessary for every time the machine is turned on initially. After this,

the only operational procedure for MTConnect is through the application to connect to the agent.

The user must also set up the data acquisition system for the external sensors. The sensors are plugged into the appropriate CompactRIO module for its type. For example, the thermocouple wires that will sense the tool temperature should be plugged into the channels of the Thermocouple module (NI 9211) of the CompactRIO, while the force sensor wires that run through the signal amplifier should be plugged into the channels of the Analog Input differential module (NI 9215). If other sensors are required for this application, the user would need to determine which module is appropriate for the type of sensor that would be used. After the sensors are wired into the CompactRIO, the user connects the Ethernet cable from the CompactRIO to the host PC. Using NI Measurement & Automation Explorer (NI-MAX), the user can identify the CompactRIO under “Remote Systems” and check that it is indeed connected, as shown in figure 5.2.

The initial setup of the CompactRIO could be time consuming at first, but once all of the appropriate drivers are installed and it is able to be connected to the PC the setup is quick and easy each additional time it is connected. Then the user must open the appropriate LabVIEW project that will run in the background. The project can be set to start automatically once the project is opened so that it would not require any additional setup. After the project is opened, the appropriate VIs must be opened and run. This normally would include at least the Real-Time VI, shown in figure 5.3, which would automatically run the FPGA code on the CompactRIO to acquire the signals on the FPGA

level. The Host VI could be used if additional data recording capabilities are required for a more in-depth mathematical analysis.

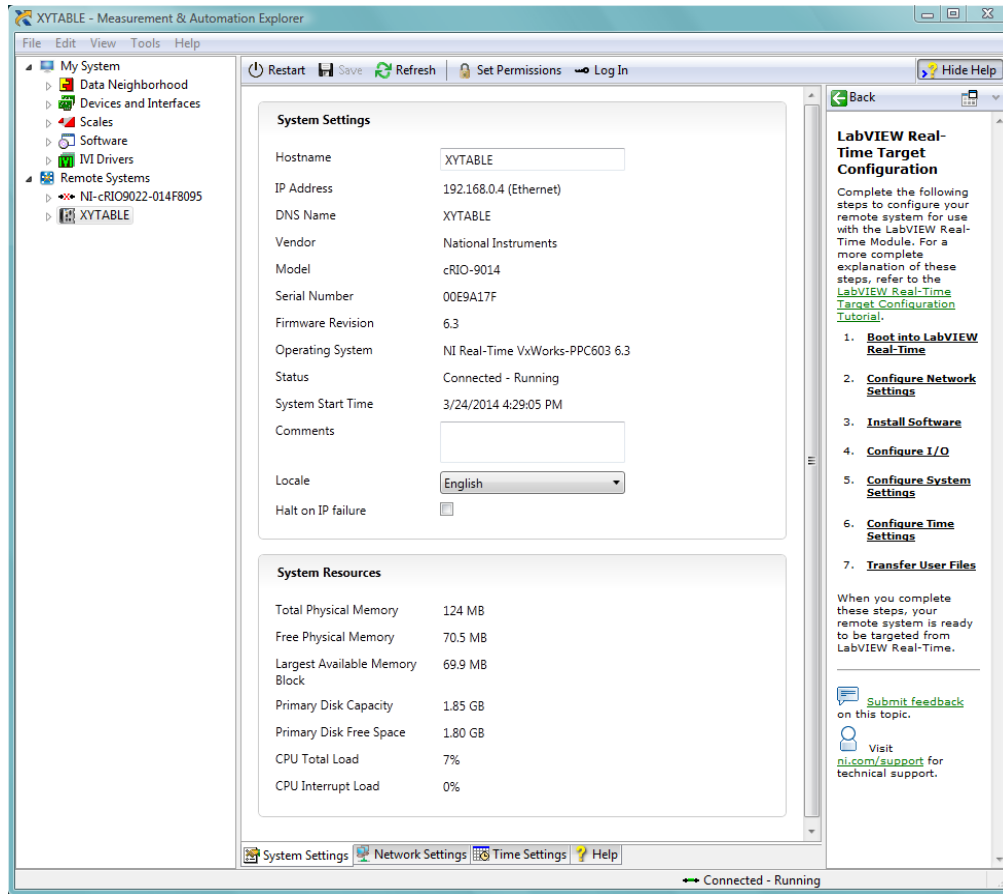


Figure 5.2 – NI-MAX showing the connection to the cRIO chassis named “XYTABLE”.

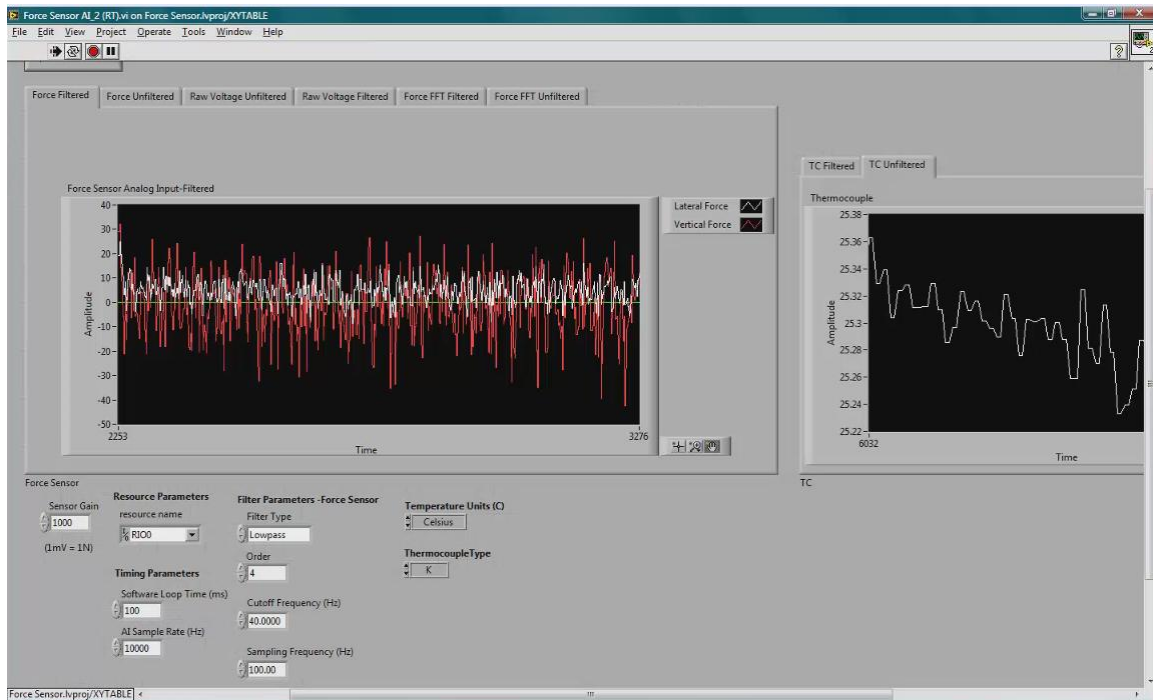


Figure 5.3 – A screenshot of the background LabVIEW Real-Time VI that is required for the external sensor data acquisition.

The user can then check the NI Distributed System Manager (found under National Instruments in the programs on the PC) to see if the shared network variables are being hosted on the server created by the NI Shared Variable Engine and that the values are changing appropriately. The user would need to make sure that these events occur before the application starts up, or to start the background elements whenever this specific data is required. If the shared variables contain data, this would indicate that the application would be able to read the shared variable data and display it on the Measurement Studio controls and indicators, which will occur at any time while the application is running.

User-Interface Operation

After the equipment is set up, the user would then start the central machine monitoring application that has been developed, as seen in the screenshot in figure 5.4. A schematic of the application's operational procedure is shown in figure 5.5 to identify the order in which the application's elements and functions will be executed by the user.

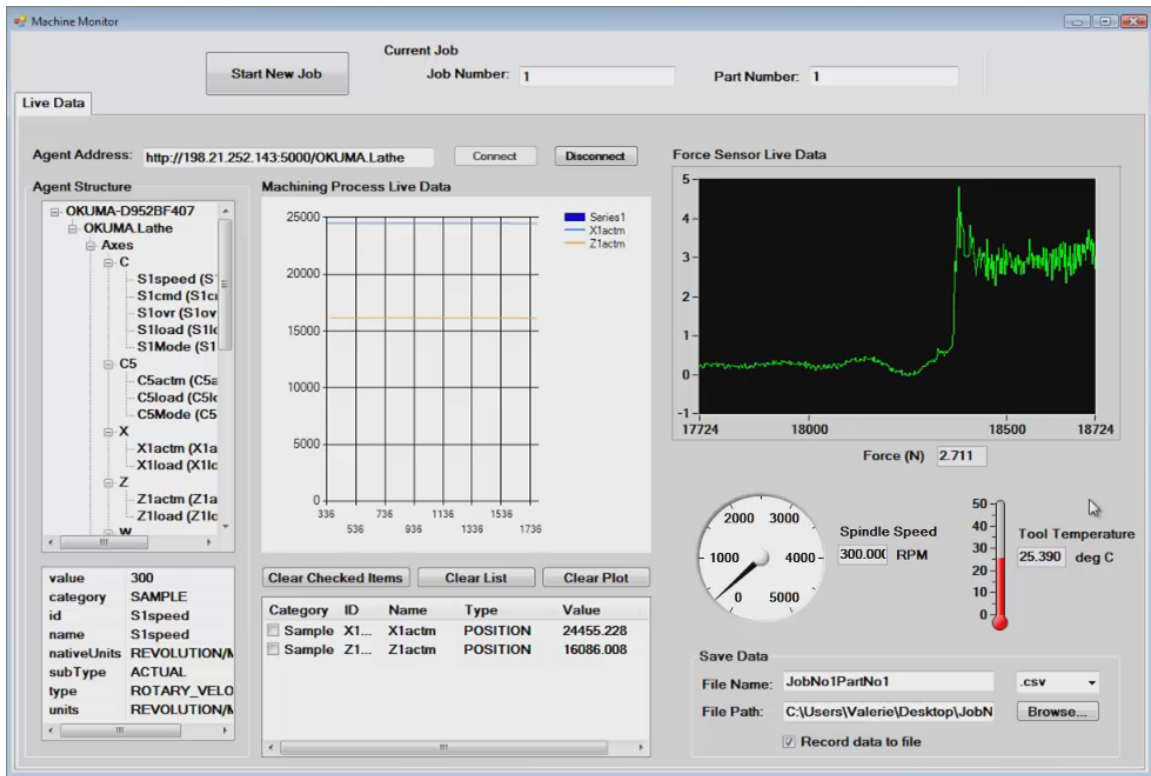


Figure 5.4 – Screenshot of application running during machining operation.

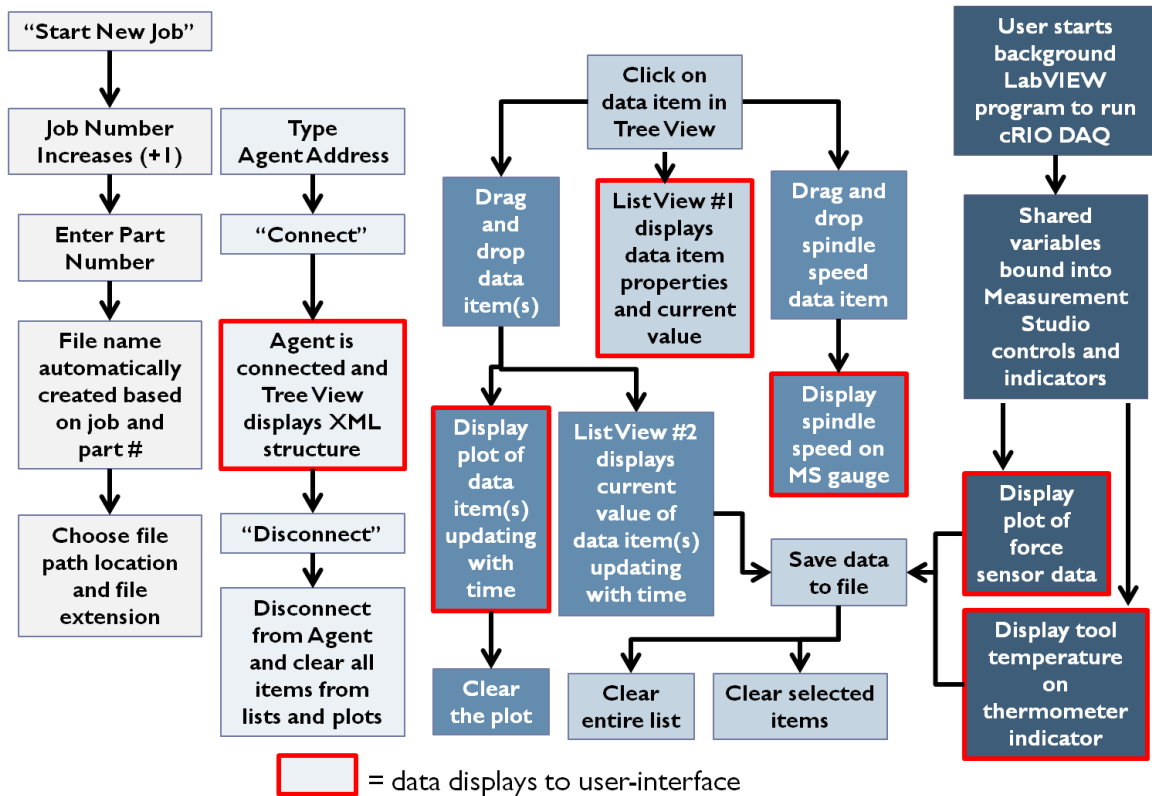


Figure 5.5 – Schematic of the application’s operational procedures.

The operation of the application’s user interface, according to figure 5.5, begins with the user clicking the “Start New Job” button. This initiates a function that increases the “Job Number”, which initially starts at 0 and increases by 1 every time the “Start New Job” button is pressed. This allows the application to continue to run and keep track of how many jobs have been performed and lets the machine operator begin new jobs without restarting the application each time a new job is to be started. The job number would appear in the “Job Number” text box on the user interface. Next, a message box pops up to prompt the user to scan or enter the part number. If a barcode scanner is used, similarly to the GE Work Setup application the scanner would be configured as a

“keyboard” input and would simply “type” the number into the message box. The user should click OK if the message box does not automatically close. The part number would then appear in the “Part Number” text box on the user interface. Screenshots of these parts of the application during this process can be seen in figures 5.6 and 5.7.

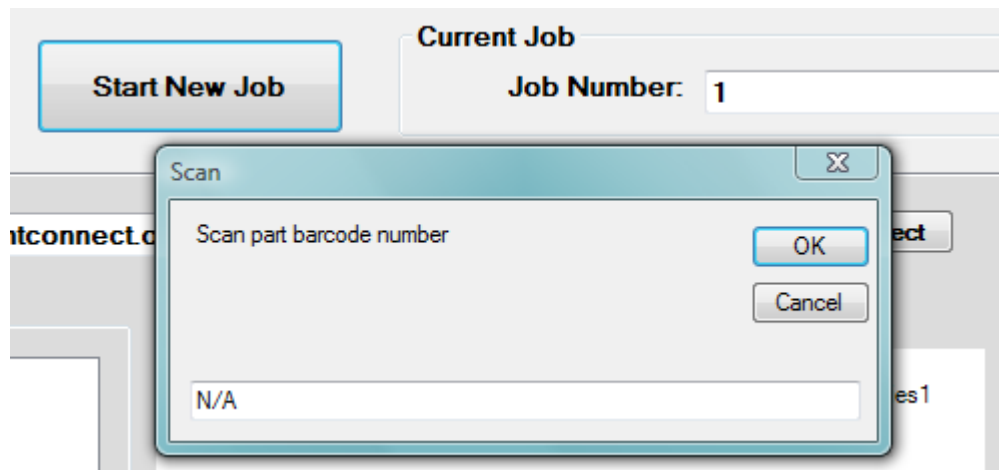


Figure 5.6 – After the “Start New Job” button is pressed, the job number is increased and shown in the text box (this would be Job Number 1). A message box pops up to prompt the user to scan the part barcode.

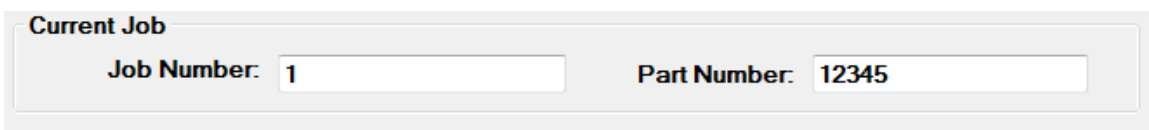
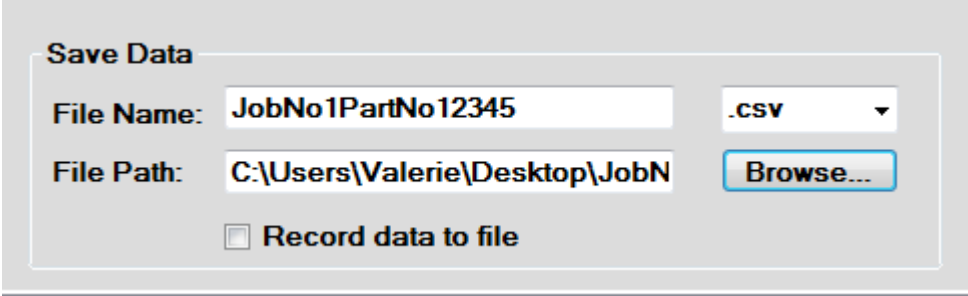


Figure 5.7 – The job number and part number, which is entered after the part barcode is scanned, for the current job.

After both the job number and part number are entered into the appropriate text boxes, a file name is created in the “Save Data” section, shown in figure 5.8, in the format “JobNoXPartNoX” with X being the appropriate job and part numbers entered into the text boxes. This occurs so that the user does not have to type in a name for the

file manually and so the job number and part number will be saved with the appropriate machining process information associated with the part being machined. The user would then choose the desired file path location and file extension (a choice of “.txt” or “.csv” have been set up in the application, but ideally any file extension could be a choice to use if it were configured in the code of the application). A message box will pop up, shown in figure 5.9, once the file path has been chosen to show the operator where the file will be saved. Once the work setup procedures are completed, the user can then begin the data collection process.



The image shows a 'Save Data' dialog box with the following elements:

- File Name:** JobNo1PartNo12345
- File Extension:** .csv (selected from a dropdown menu)
- File Path:** C:\Users\Valerie\Desktop\JobN
- File Path Action:** Browse... button
- Record data to file:** (unchecked)

Figure 5.8 – The “Save Data” section contains the automatically created file name, user chosen file extension (shown here as “.csv”) and the chosen file path. The user would click the “Browse...” button to choose the file location. When ready, the user can check the “Record data to file” box to save the data, and uncheck to stop saving data.

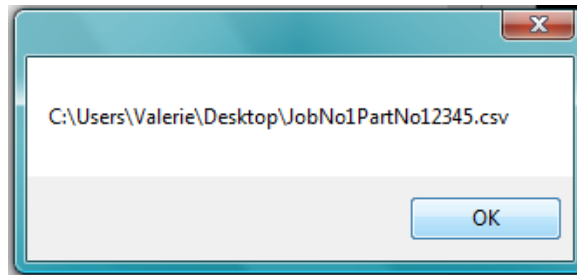


Figure 5.9 – This message box pops up to notify the user of the chosen file path location.

The next procedure begins with typing the agent address into the “Agent Address” text box, and then clicking the “Connect” button, shown in figure 5.10. This initiates the same code from the MTConnect Sample Client to connect to the desired MTConnect agent. The same procedure occurs when the agent is connected which sends the Probe command to the agent to fill the “Agent Structure” tree view with the device’s available components and associated data items such as the machine axes positions in the X, Y, and Z direction, the spindle speed, feed rate, and other important machining process information, as seen in figure 5.11.

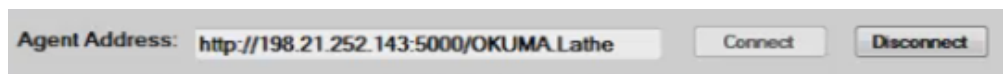


Figure 5.10 – The agent address of the Okuma LB4000EX machine tool is entered into the text box and the “Connect” button is clicked to connect to the agent. The “Disconnect” button can be pressed to disconnect from the agent and clear all lists and plots.

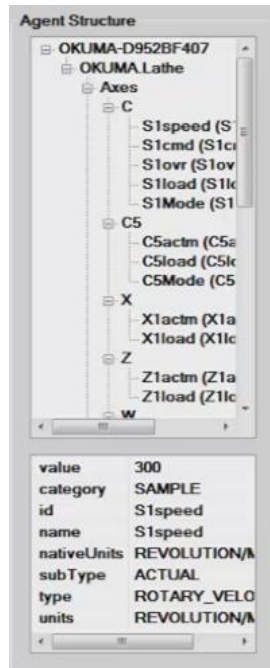


Figure 5.11 – This tree view (top) shows the structure of the MTConnect agent that retrieves data from the adapter on the Okuma LB4000EX machine tool. The list view (below) shows a specific data item’s properties and value. This specific data item illustrated is the spindle speed with a value of 300 and units of “revolution/minute”.

At this point, the user can choose to disconnect from the agent by clicking the “Disconnect” button which would clear the data from the tree view, list views, and plot that consists of MTConnect data. This would be appropriate if the operator wants to connect to a different agent, reconnect to the desired agent, or shut the application down for that shift.

If the user wants to remain connected to the agent, they can click on individual data items and their values will show up in the list view below the tree view (list view #1), showing the specific data item’s properties and value which will change according to the same DataItemWatcher function that was mentioned previously as a part of the MTConnect Sample Client. This also initiates the “Sample” command to the agent. The

values will change in near-real time and the data items' properties will be displayed as well, including the appropriate units of measurement for each data item.

The user could choose to drag and drop the data item into a few places. If the user wants to plot the data item, the user would click and drag the data item into the onto the “Machining Process Live Data” plot, shown in figure 5.12. Several data items can be dragged into the plot and would display as separate series updating at the same rate according to the DataItemWatcher function. Also, the user could click and drag the data item into another list view below the plot (list view #2) so that they could prepare to save any combination of data items desired. The application could record the value of each data item that is listed in list view #2 in a column of the file previously defined.

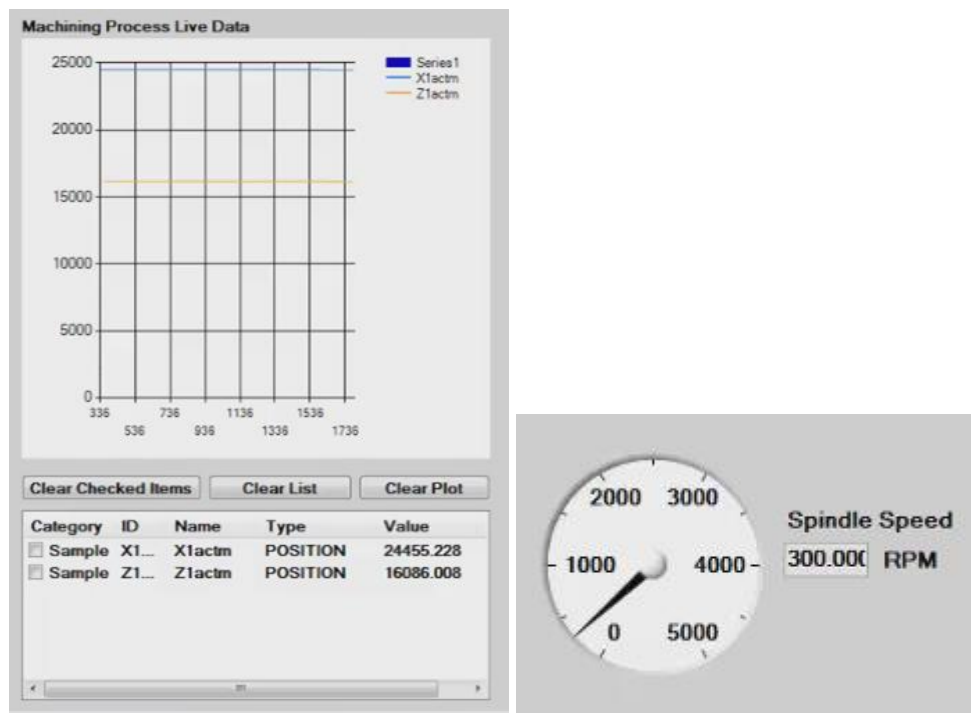


Figure 5.12 – MTConnect data items can be dragged into the “Machining Process Live Data” plot (top left) and the list view (bottom left) as well as the “Spindle Speed” gauge (right), which is a Measurement Studio indicator, to display the appropriate values.

The user could also click and drag the spindle speed data item into the “Spindle Speed” gauge, which is a Measurement Studio indicator. If the user drags and drops the spindle speed data item the gauge will update to show the correct value of the spindle RPM. The user can technically drag and drop any data item because the gauge is programmed to accept any data item, but the only useful data item for the setup and range of this gauge is the spindle speed. The data displayed in the gauge will not be saved directly, but can be saved if this same data item is dragged into list view #2.

If the operator desired, at this point they could clear the entire plot by clicking on the “Clear Plot” button, clear the entire list view #2 by clicking on “Clear List”, or clear selected data items from list view #2 by clicking on the data item’s check box next to its name and then clicking “Clear Selected Items”. This would be useful if the user wants to look at different data item values, save a different combination of data items, or clear everything and start over.

As previously mentioned, the force sensor data automatically displays after the application is opened and starts running. The data from the appropriate shared variable is bound to the Measurement Studio waveform graph, “Force Sensor Live Data”, as well as an associated “numeric edit” indicator below the plot which shows the current value of the force, as seen in figure 5.13. The CompactRIO can be set up to acquire data at a certain sampling rate through LabVIEW, but due to the application updating the MTConnect data at a set 100 milliseconds this can delay the updating of the external sensor data slightly. The data acquisition sampling rate would still be the same, but the graph may appear to update at a slower rate due to PC processing limitations. Also, the

force sensor signal is set to record to the same file as the MTConnect data items from List View #2. However the data recording frequency is dependent on the updating frequency of the MTConnect data items, so the force sensor signals will only record at that same frequency regardless of the sampling rate defined in the LabVIEW programs. If more data points are required for a more in-depth numerical analysis, the user should record the force sensor data through LabVIEW as well as through the application. In addition to the force sensor data, the tool temperature can be read through the Measurement Studio thermometer indicator as seen in figure 5.14, which is bound to the shared variable containing the thermocouple signal. The associated “numeric edit” indicator next to the thermometer indicator shows the value of the thermocouple reading to the user. This will also update in the application in the same way as the force sensor, and is also set to record to the same file in a separate column.

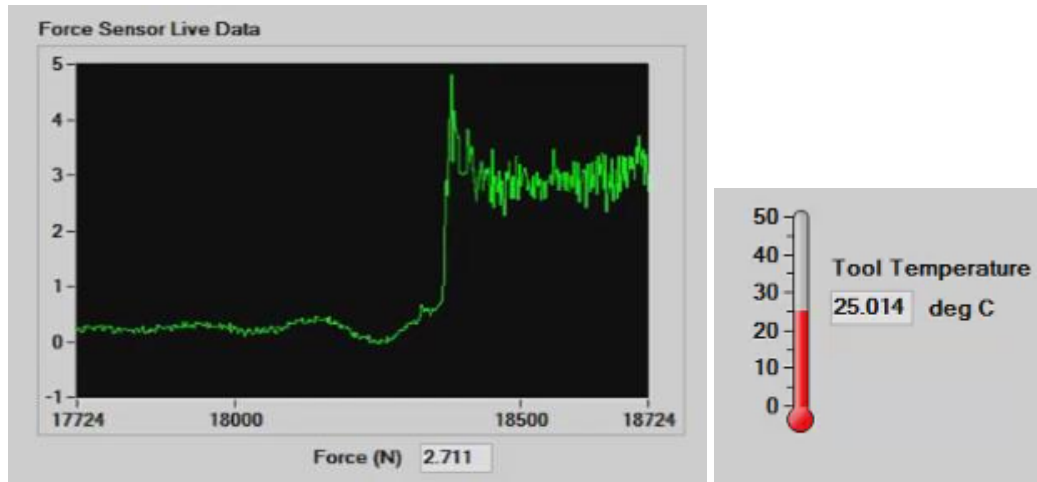


Figure 5.13 – The “Force Sensor Live Data” waveform graph and the “Tool Temperature” thermometer are Measurement Studio indicators that accept data from the shared variables originating from the background LabVIEW program connecting to the CompactRIO.

When the user is ready to save all of this information (the chosen MTConnect data items, the force sensor data, and the thermocouple data), they would click the “Record data to file” checkbox which will initiate the data recording events. The data will record to the file as long as the box remains checked. Once the user is ready to stop data collection, they would have to uncheck the box. Again, the user must be aware that the application will only record the external sensor data at the rate at which the MTConnect data items are being recorded in order for the information could be saved in the same file. If the user wanted more data from the external sensors that might be more appropriate for some mathematical post analysis, they would benefit more from saving the data through LabVIEW, which is an easy procedure to set up. The file types that were written into the application for the user to choose, either a .txt or a .csv, can be imported into many types of analysis software packages including Microsoft Excel and Matlab. An example of the

file opened in Excel is shown in figure 5.14, where the data can be plotted to show the change in machining conditions over time for specific parts.

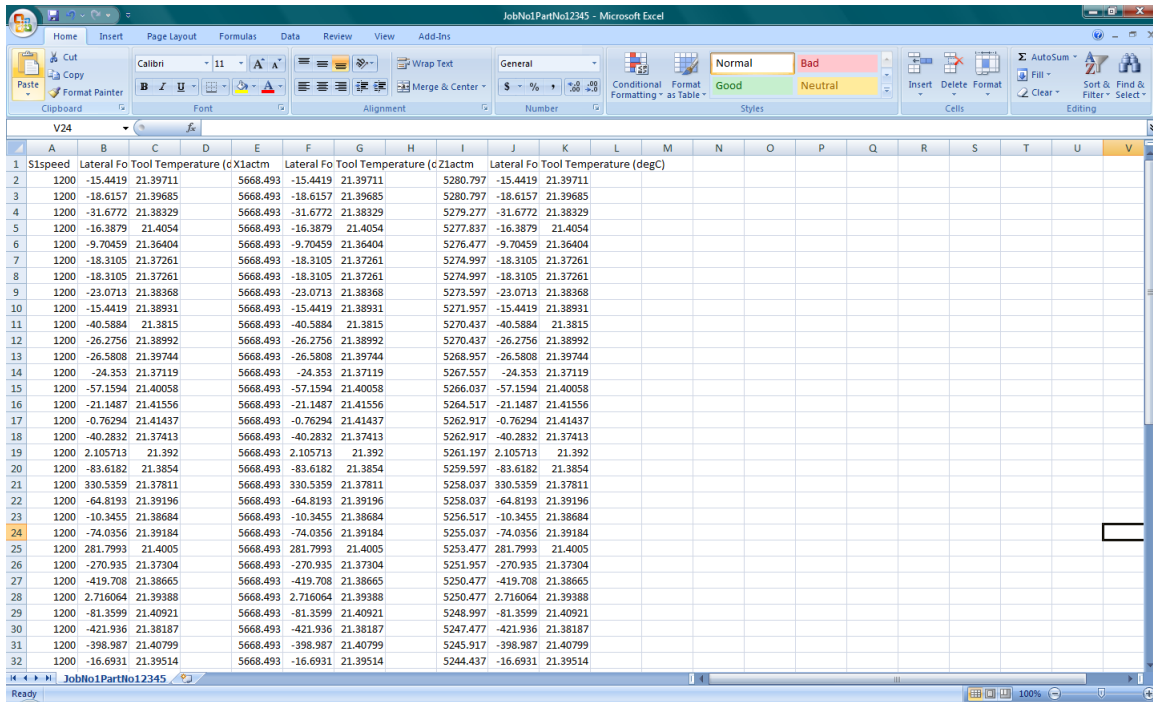


Figure 5.14 – A screenshot of the .csv file in Microsoft Excel with data from the machining process collected through the application. The user can save any combination of data items along with data collected from each sensor and can plot this data for analysis. The filename reflects the job number and specific part number so the user can keep track of the part-specific machining process data.

After a cutting operation is completed, the user could clear the information or disconnect from the agent as previously mentioned. If the user wanted to start a new cutting operation, they would identify the start of a new job by clicking the “Start New Job” button again which would increase the “Job Number” and allow the user to enter or scan another part number into the system. The user could choose to disconnect and reconnect to the agent, clear the plot and list boxes manually, or leave the MTConnect

agent running through the application while the new part was being loaded into the machine. This should not affect the functionality of the application in any way.

The extensions to the Sample Client application were added for more convenient use of the application for multiple jobs instead of being a “one-time use” application that needed to be closed in order to be cleared of the data items and reset to its original form. It also incorporates the part tracking functionality of the GE Work Setup application and allows the user to save important machining process data according to the specific job and part numbers. The application not only allows the user to save data but it provides a near real-time monitor of the machining operation which, with the assistance of sensor information, will aid in identifying machining conditions that could be detrimental to part quality.

CHAPTER SIX

APPLICATION IMPLEMENTATION FOR CHATTER IDENTIFICATION

This chapter will address how the application can be implemented to meet different needs of the manufacturing industry, including using it as a tool for chatter identification and to use it to track part-specific information for better part quality and productivity in the machining process.

Chatter Identification Techniques

As discussed in chapter 2, chatter is caused by a variable chip thickness during cutting which leads to a variable cutting force. With the use of a force sensor, the cutting force components can be detected and used to identify chatter conditions on-line using monitoring applications, such as the one that has been developed. This is a more practical technique of identifying chatter in an industrial manufacturing facility than the use of a stability lobe diagram and its associated frequency response function.

The chatter identification software application that has been developed can display the force sensor signals to monitor for chatter conditions along with displaying the feed rate, spindle speed, axes positions, etc. through MTConnect to identify parameters that could have caused chatter in the machining process. From there, based on the process dynamics, the user can identify the minimum force variations that would be acceptable before chatter is to occur in the process. A spectral analysis can also be performed to identify the frequencies of vibration that have large amplitude peaks, indicating a large variation in force and therefore possible chatter conditions.

Chatter Simulation

A simulation of a machining chatter condition was performed to test the functionality of the chatter identification application. The ability to perform live machining became unavailable after Okuma decommissioned the machine to be sold to one of their customers and sent a replacement machine to the laboratory at CU-ICAR to be commissioned at a later date. Also, a simulator system can be taken to the MTConnect “MC2” Conference, where the application can be demonstrated during the final phase of the competition.

The chatter simulation consists of placing the force sensor and thermocouple integrated tool (which was removed from the machine prior to decommissioning) on a vibration shaker table [81], as seen in figure 6.1. This induces vibrations that the sensor can pick up and display in the application as variations in force. The vibration shaker table vibrates by displacing the table up and down at 60 Hz with a knob to vary the strength of the vibrations.



Figure 6.1 – Vibration shaker table. [81]

This system simulates the tool vibrating at a natural frequency of the tool-workpiece interaction which, according to [1], results in chatter on the surface of the part. The background LabVIEW program reads the signals from the force sensor, which shows large variations in the force due to the vibrations, thus mimicking the chatter phenomenon. Also, the LabVIEW program performs a spectral analysis to calculate and display the single-sided (absolute value) amplitude spectrum in the frequency domain to determine the frequencies at which the system vibrates (shown as peaks in the amplitude at specific frequencies). These signals are saved in shared variables and read by the chatter identification application in near-real time. The application detects the simulated chatter and displays machining parameters to the user through MTConnect, allowing the user to determine where chatter occurred on the part and what machining parameters could have caused it. The user can then use that information to adjust the machining parameters accordingly to suppress chatter and reduce further damage to the part. The application could be programmed to perform the chatter suppression automatically if the MTConnect standard was “Read/Write” instead of “Read Only”. This would allow the parameters to be adjusted and controlled through the application and sent back to the machine controller by writing to the MTConnect agent.

Spectral Analysis in LabVIEW

The spectral analysis of the vibrating system is performed in LabVIEW to reduce the computational time needed to perform the calculation in the code of the chatter identification application. The ability to perform this analysis exists as well in the Measurement Studio toolkit for Visual Studio, however it is more complex to program

and requires knowledge of the functions, code classes, methods, and libraries and there is limited documentation on how to use these for live data acquisition applications. Therefore, performing the calculations in LabVIEW and sending the data through the shared variables into the application was the process chosen for this analysis.

The block diagram of the LabVIEW VI on the Real-Time level, as seen in figure 6.2, shows the programming behind the spectral analysis. The “Amplitude and Phase Spectrum PtbyPt” sub-VI computes the Fast Fourier Transform (FFT) of the time-domain signal sent in one data point at a time (of the double precision floating point data type) and outputs an array of double precision floating point data types as the magnitude of the spectrum. The array can be clustered with the df output, which is the frequency resolution, and the constant 0 to plot the magnitude array against a range of frequencies in LabVIEW for troubleshooting. The magnitude array is bound to the “ForceFFT_amp” shared variable which then can be read into the chatter identification application through the same type of connection as previously defined for the force sensor time-domain signal. The only difference is that the time-domain signal, since the data is read one double precision floating data point at a time into the application, uses the “PlotYAppend” data binding to the Measurement Studio waveform graph while the frequency-domain signal, which comes in as an array of double precision floating points and therefore uses the “PlotY” data binding to another Measurement Studio waveform graph. The difference is that the “PlotYAppend” property appends the next data point of the signal onto the plot after the previous data point, while the “PlotY” property waits for a certain number of samples, depending on the defined sample length, to update the plot.

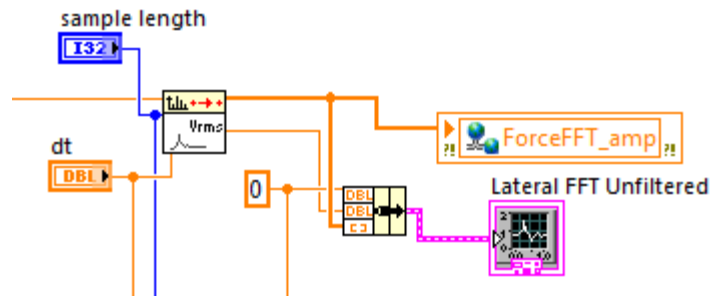


Figure 6.2 – Amplitude and Phase Spectrum PtbyPt sub-VI used in the spectral analysis of the force sensor.

A few tests were performed to determine the optimal sample length for a high resolution spectral plot while considering update time for the application. The tests were performed with a period dt of 0.001 seconds input to the amplitude spectrum sub-VI which depends on the sampling rate defined for the system. The sampling rate for the system on the FPGA level was 10,000 Hz, but since the software loop time on the Real-Time level was set up as 100 milliseconds, the samples per update of the loop gave a sampling rate on the Real-Time level as 1000 Hz. Therefore, the period for the amplitude spectrum sub-VI was set as 0.001 seconds, or 1 millisecond to compensate for that.

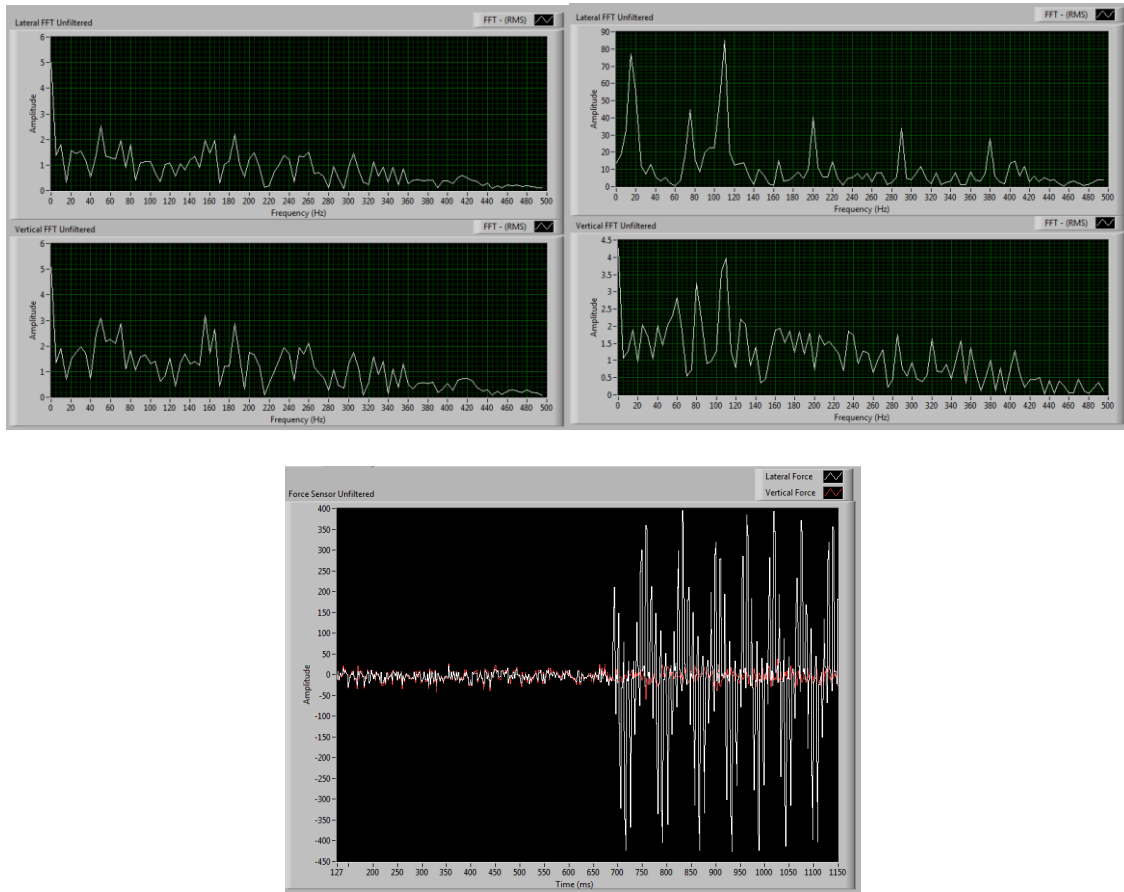


Figure 6.3 – The single-sided amplitude spectrum plot with a sample length of 200 (top left: normal conditions, top right: chatter conditions) from the unfiltered force sensor signal (bottom). Chatter was simulated on the force sensor as seen by the large variations in force at approximately 0.7 seconds, after which the amplitude spectrum sub-VI updated and displayed the frequency domain signal on the plot.

The first test was for a sample length of 200 input to the amplitude spectrum sub-VI. The program must wait for 200 samples to be collected to calculate and display the single-sided amplitude spectrum in the frequency domain. Because of this, there is a period of time in which the spectrum will have low resolution until it is able to gather enough data points to represent the spectrum with the entire number of samples defined

and therefore only outputs near-real-time data. This is shown in figure 6.3 along with the associated time-domain signal.

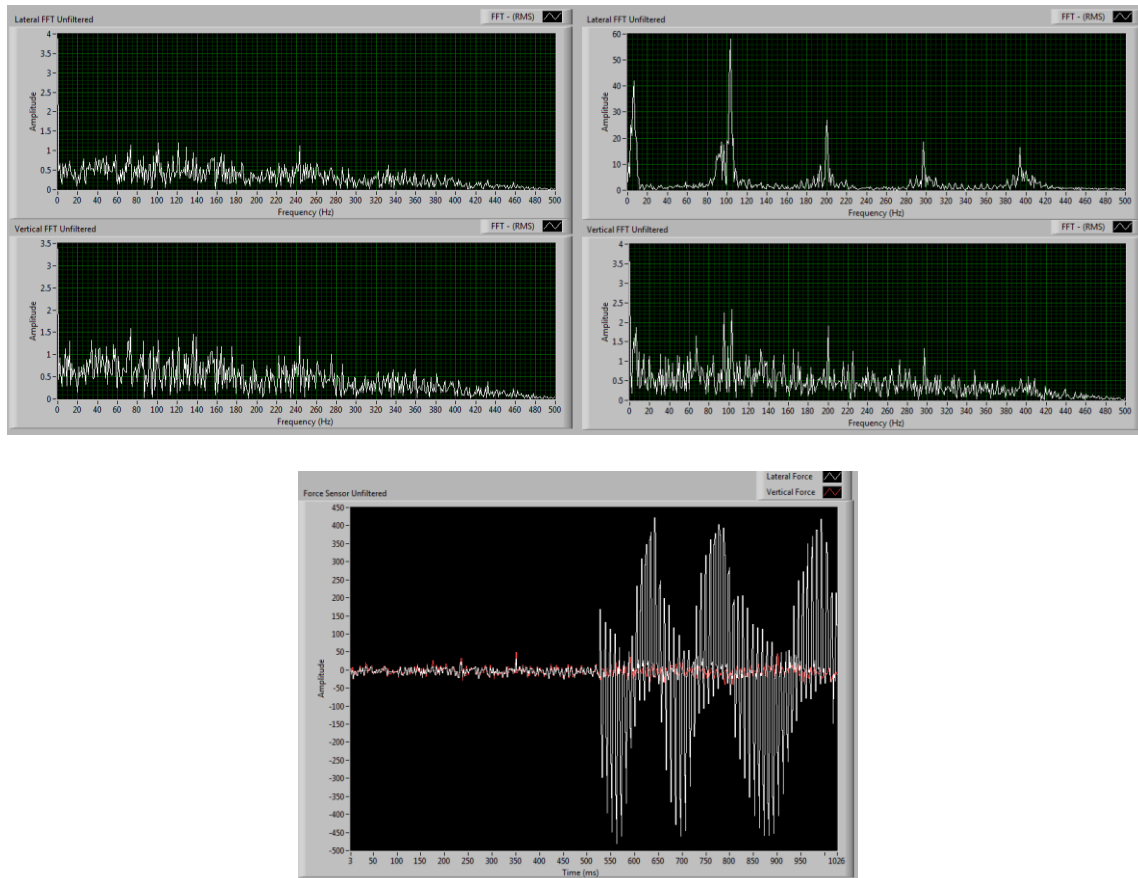


Figure 6.4 – The single-sided amplitude spectrum plot with a sample length of 1000 (top left: normal conditions, top right: chatter conditions) from the unfiltered force sensor signal (bottom). Chatter was simulated on the force sensor as seen by the large variations in force at approximately 0.5 seconds, after which the amplitude spectrum sub-VI updated and displayed the frequency domain signal on the plot.

A larger sample length was tested to increase the resolution of the peaks at the natural frequencies of the system. For a sample length of 1000 samples, the program must wait a longer amount of time to be able to collect 1000 samples to update the calculation

and the plot. However, the resolution of the frequency spectrum is much higher than that of 200 samples, as shown in figure 6.4. A good compromise between a relatively high resolution spectrum plot and a relatively fast update time was found at a sample length of 500 and can be seen in figure 6.5. This shows relatively high resolution at the peaks of the frequencies of vibration and updates at a rate which allows for a better approximation of “real-time” data acquisition.

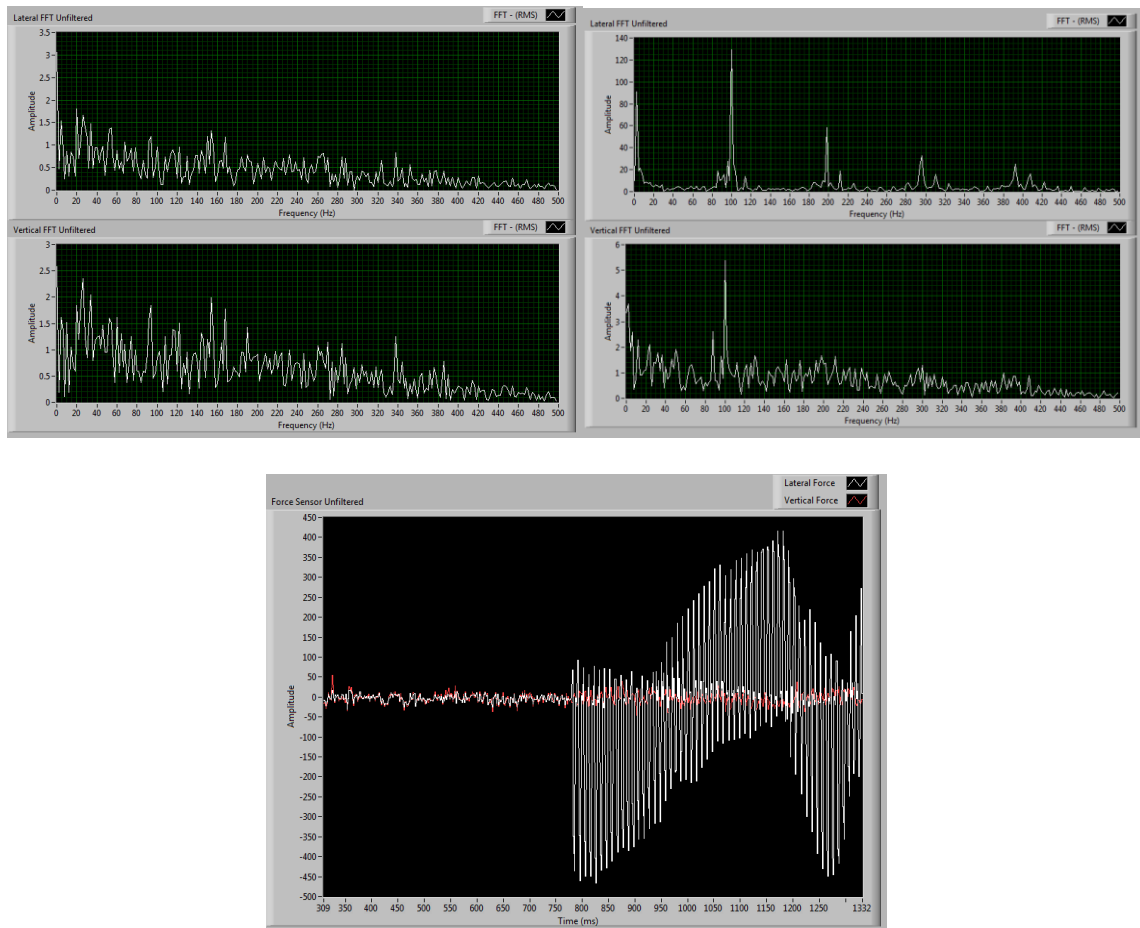


Figure 6.5 – The single-sided amplitude spectrum plot with a sample length of 500 (top left: normal conditions, top right: chatter conditions) from the unfiltered force sensor signal (bottom). Chatter was simulated on the force sensor as seen by the large variations in force at approximately 0.8 seconds, after which the amplitude spectrum sub-VI updated and displayed the frequency domain signal on the plot.

After the final choice of parameters for the single-sided amplitude spectral analysis of the force sensor signals, the data output from the amplitude spectrum sub-VI was sent to a shared variable on the Real-Time level and then bound to a shared variable on the Host PC level to be transmitted to the chatter identification application. It is there that the chatter simulation would be also performed to test the capabilities of the final application.

Figures 6.3 through 6.5 show a dominant amplitude peak at 100 Hz and its associated harmonics. A few reasons for the discrepancy between the results shown on the plots and the actual frequency applied to the tool are presumed to be the following. Since the point-by-point amplitude spectrum sub-VI in LabVIEW was used to calculate the FFT takes in data points one at a time, the calculation is delayed and could alter the results. A more accurate approach would be to perform the FFT on a large array of samples at a time instead of one data point at a time; however this could sacrifice the update time in LabVIEW and in the application. Also, the FFT could be performed on the FPGA level for a more accurate calculation. The reason this was not done originally was due to the large compile time of the FPGA. If a user of this system requires more accurate frequency domain results, this would be the best approach.

Another reason for the discrepancy in frequency could be that because the tool is not rigidly secured to the vibration shaker table, the vibrations of the tool and the table are coupled, resulting in a peak at 100 Hz instead of 60Hz. In a few of the plots, a low frequency component of around 5 Hz can be seen, which could either be the DC offset or

a low frequency vibration due to the lack of a secure attachment of the tool to the vibration table.

Results in Chatter Identification Application

The chatter simulation was also performed using the chatter identification application to detail its feasibility in detecting chatter conditions. The data acquisition equipment, the integrated tooling sensor, and the vibrator shaker table were brought to the Okuma North America headquarters in Charlotte, NC to integrate with their machine simulator on an OSP-P300 controller. This collaboration was done in preparation for the MTConnect “MC2” Conference, where this application will be presented for the final round of judging. While at the Okuma facility, the simulation was done while connected to an MTConnect agent on the machining simulator which ran a program that simulated a straight cut at a specified spindle speed and feedrate.

Using the same optimal filtering and frequency spectrum parameters mentioned previously, the force sensor detected the simulated chatter in the application while acquiring machining parameter values from the MTConnect agent. Figure 6.6 shows the “Lateral Force” tab open in the “Force Sensor Live Data” plot area, displaying a normal condition (small variations in force) leading into a chatter condition (large variations in force). The same figure also shows the “Lateral FFT” tab open which shows the single-sided amplitude spectrum of the “chatter” condition. Figure 6.7 shows the same plots for the vertical force signal.

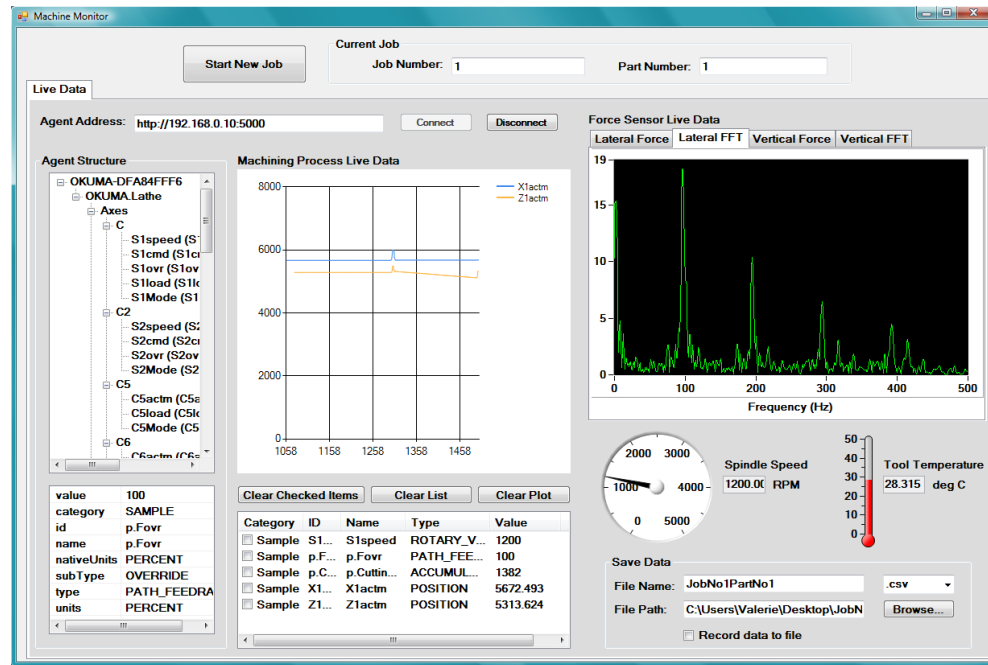


Figure 6.6 – The chatter identification application connected to the MTConnect agent on the Okuma machine simulator and the CompactRIO to acquire the force sensor data. The top figure shows the lateral force signal in the time domain, where the bottom shows the lateral force signal in the frequency domain.

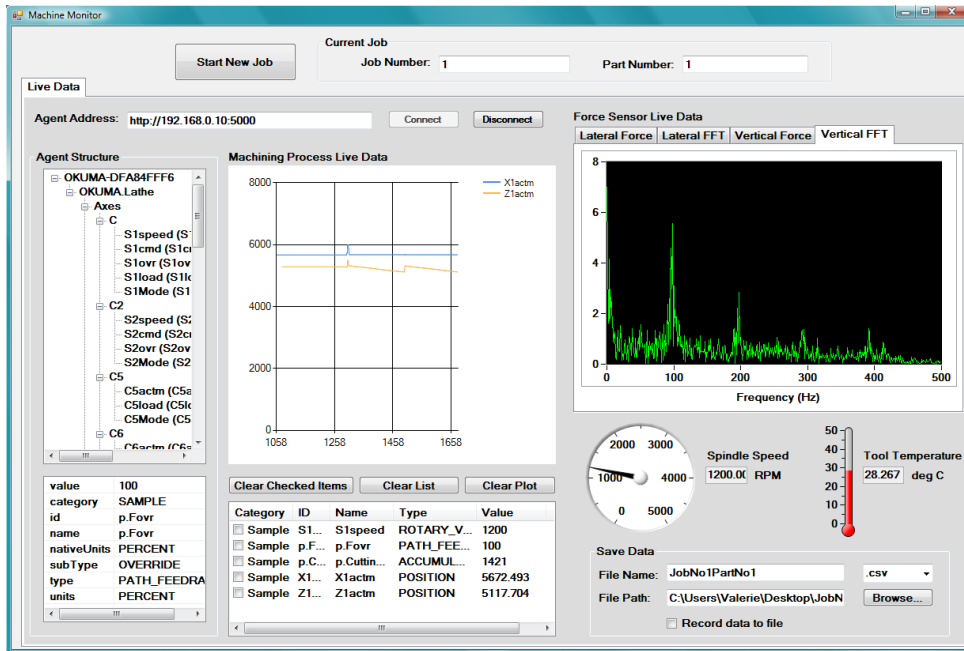


Figure 6.7 – The chatter identification application connected to the MTConnect agent on the Okuma machine simulator and the CompactRIO to acquire the force sensor data. The top figure shows the vertical force signal in the time domain, where the bottom shows the vertical force signal in the frequency domain.

It is also useful to note that the lateral force strain gauges show greater amplitude of vibrations than the vertical force strain gauges regardless of the position of the sensor on the shaker table. Various orientations were attempted: the lateral gauges were placed along the direction of motion of the table vibrations while the vertical gauges were orthogonal to the vibrations, and then the vertical gauges were placed along the direction of the vibrations while the lateral gauges were orthogonal to the vibrations. Regardless of the orientation, the plots showed the same results: the lateral force signal always had larger amplitudes of vibrations than the vertical force signal. This could be due to the strain gauge construction or the routing of the signal through the amplifier. It is possible that the amplifier gains were adjusted at some point before the analysis. However, care was taken to only adjust the zero setting and not the gain setting which was set during the initial calibration procedure of the sensor. For future implementations of this sensor, a more sophisticated calibration procedure should be done using a force dynamometer for better accuracy, as well as checking the amplifier settings and the strain gauge construction on the tool. Also, future implementations should consider programming most of the complex calculations on the FPGA level to increase the accuracy and precision of the measurements if this is required for experimentation.

The data that can be saved through the application can be opened in Microsoft Excel for future analysis of the process. Figure 6.8 shows the data in the file for each data item that was saved along with the data from the sensors. The user can plot this data to show the variations in the force signal that could indicate a chatter condition. The user can also plot the tool position, such as the Z-axis position, to identify where chatter could

have occurred on the part. Other data items could be saved as well to indicate what parameters could have caused the condition to occur. All of this data is important in the analysis of the chatter condition in the machining process and is saved according to the specific part number and job number of the operation to keep track of this information for the analysis.

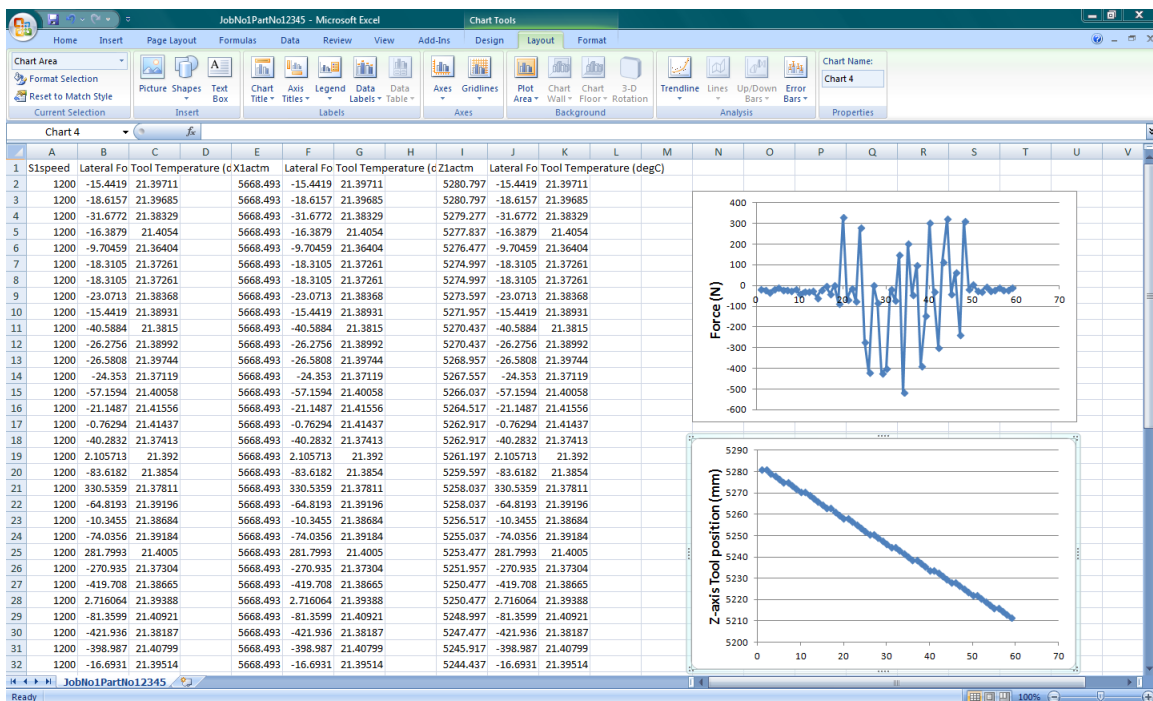


Figure 6.8 –The screenshot of the .csv file opened in Excel shows the force variations in the top plot and the Z-axis position of the tool in the bottom plot. The user can identify where chatter occurred as the tool moved in the Z-direction on the part. The user can also look at other data items that were saved to identify what parameters could have caused the chatter condition.

This simulation shows that the application is capable of accepting data types from a spectral analysis in LabVIEW using shared network variables. The calculations for the spectral analysis could also be hard-coded into the application in Visual Studio with the

help of the Measurement Studio toolkit. This would be necessary if the sensor signals could be routed through the machine controller and sent to the application through the MTConnect agent if its sensor definition was written into the agent code. The spectral analysis in Visual Studio/Measurement Studio would require extensive knowledge of the functions, classes, objects, methods, etc. and can be complex for a non-programmer especially since there is limited documentation about the Measurement Studio toolkit capabilities.

The simulation at the Okuma facility was successful in integrating the two systems through the chatter identification application. It was also successful at demonstrating that the combination of data from the machine tool controller through MTConnect and the data from a force sensor through a proprietary controller could be used to detect tool vibrations that could signify chatter at specific tool positions. Also, the machining parameters such as the spindle speed and feed rate are shown along with the force data which could be used to identify what parameters could have caused the chatter conditions and what parameters would need to be adjusted for chatter avoidance in future operations.

CHAPTER SEVEN

RECOMMENDATIONS FOR FUTURE WORK

This chapter will discuss the limitations and shortcomings of the MTConnect standard and the data acquisition tools used, and then discuss recommendations for future work based on the long-term implementation vision that can be seen for this system and its associated software application. In addition, the benefits of this application to the manufacturing industry will be presented.

Limitations and Shortcomings

MTConnect Limitations

There are a few limitations to the MTConnect standard and communication process that prevented a few of the original ideas for this application from being implemented. The first limitation is the ability to integrate sensors through MTConnect. It was originally thought, based on the background research done to learn about MTConnect, that the addition of *assets* to the standard would allow the integration of sensors to be used and communicated through MTConnect components. *Assets* are defined as something that is not a component of the device but can be removed without detriment to the device and can be used to acquire and transmit data through the MTConnect standard [6]. The documentation claimed that the addition of assets would be included in the next version of MTConnect. It was assumed that this would be the definition required to integrate sensors into the standard and that it was impossible to do so without this definition. However, while discussing this application with a few

members of the MTConnect Institute during a web conference for the MTConnect Challenge, they mentioned that there are thousands of different sensor types defined in the MTConnect standard. This would require the definitions to be written in the adapter code to translate the sensor data from its proprietary format to the MTConnect-compliant format, and it would require the agent code to be written for that translation to be allowed. However, since the sensor was custom-built by students at the university and is not an Okuma supported sensor, it would have had to be customized into the Okuma adapter and agent code explicitly as additional devices or components under the machine tool device definition. Also, additional data acquisition hardware capabilities on the Okuma machine tool would have been necessary to read the sensor signals and process them through the machine tool controller and then through MTConnect. This could be a potential future addition that would help extend the functionality of this application and make it easier to use without the peripheral data acquisition equipment and software.

Another shortcoming of the MTConnect standard is the Read-Only nature as the standard exists currently. This means that the data collected from the machine tool can only be read through the MTConnect standard to a client application but cannot be written back to the machine tool controller through the MTConnect agent. This limits the ability to perform control applications through MTConnect which means that the control of the machine tool must exist externally. If the MTConnect standard were Read-Write, meaning the read ability would still exist and the writing capabilities to the agent would be included, this would allow for easier control applications since the data communication aspect would be much simpler. The user of the application that could

connect to the MTConnect agent would simply enter values of certain parameters that would need to be controlled, such as the position of an axis of the turret or the value of the feed rate, or the application could perform these control operations automatically by using desired control algorithms. This would eliminate the need for proprietary hardware controllers external to the machine tool itself, such as the CompactRIO for example, and would therefore eliminate any background software needed to operate the controller, such as LabVIEW.

Another limitation is that the MTConnect standard, while open-source, is not freely modifiable or customizable to meet individual needs. This could be beneficial for industry because it would prevent individuals from creating their own “proprietary” standards, but it prevents academics from making customizations and extensions to the standard for improvements. Individuals must request additions or modifications to the standard through the MTConnect Institute, which is up for judgement and possible denial of the additions from the standard. MTConnect is a relatively simple definition of a machine tool’s components and the communication process is relatively simple, so this may not be a very significant limitation.

Proprietary Data Acquisition Tools

There are several limitations to data acquisition tools such as the CompactRIO and LabVIEW. The primary limitation is that they are proprietary in nature which makes it expensive to purchase the hardware and software licenses. Also, they are limited in their connection to other technologies outside of the ones supported by the specific

manufacturer. Another shortcoming is that while Measurement Studio is a helpful tool in connecting two proprietary technologies (NI LabVIEW and Microsoft Visual Studio), there is not enough clear documentation on the Measurement Studio tools and capabilities of those tools. The documentation for most NI products is very clear and helpful, and they include support from the software and application engineers as a part of the license package. A similar situation occurs for Visual Studio, as well as other Microsoft products. For both technologies, there is also an extensive user support base with forums to facilitate questions from users and answers from other users or software engineers from the particular company. However, since Measurement Studio is a relatively new software toolkit, neither National Instruments nor Microsoft has enough documentation or user feedback to assist in developing software with this tool. The National Instruments support base has a few documents on the code libraries but it requires extensive programming knowledge to understand how to use the classes, methods, objects, and functions properly. NI also provides a few simple tutorials on how to use the familiar LabVIEW controls and indicators in Visual Studio but they do not provide information on how to write the code behind the elements or how to perform more advanced operations using these elements.

Another limitation to Measurement Studio is that it only supports a few hardware driver types, such as NI-DAQmx, which does not include support for the NI-RIO driver that is used by the CompactRIO. Because of this, the connection to the cRIO was indirect through the shared network variables which required a background LabVIEW program to acquire the data from the sensor signals through the cRIO to be saved into the shared

variables. With other NI-DAQmx supported devices, Measurement Studio has a “DAQ Assistant” which is a wizard that helps connect directly to the device to display the raw signals from the device’s channels. This DAQ Assistant does not connect to the cRIO since it runs with the NI-RIO driver instead. If the NI-RIO driver was supported through Measurement Studio, the DAQ Assistant or a similar wizard could connect to the cRIO and display the raw signals from the sensors without a background LabVIEW program. All of the required data calculations and manipulations could be done on the Measurement Studio platform and all of the necessary information could be read through a single application. Also, this would allow for more users to be able to utilize the application without having to install LabVIEW to run the hardware equipment. They would only have to install the single application and connect to the external hardware if required by the system. This would eliminate the need for a background LabVIEW program regardless of the addition of the sensor definitions through MTConnect.

Proprietary Software Development Tools

Although MTConnect is an open-source standard, allowing any interested user to download the source code for the agent, adapter, and client softwares, the tools that would be required to develop client applications are not open-source. Developing MTConnect compliant applications requires some type of IDE platform which are usually proprietary in nature. This is a limiting factor for the expansion of MTConnect applications, and is unfortunately not easy to work around since many open-source software development tools have large learning curves for beginner programmers and

often do not have an easy-to-use GUI associated with it. Also, in order to develop an application on the familiar Windows platform, a Microsoft product must be used which is proprietary in nature and requires a software license. Some of their products are available as an “Express” download, which is pseudo-open-source because it allows users to download the software for free but does not include all of the software capabilities of the full version. This may be sufficient for many MTConnect applications that are developed and could help expand the usage of MTConnect in industry and development of innovative applications in academia as well.

Integration of Technologies

Some issues in integrating the different technologies used in this system were discovered while setting up the equipment and performing tests for feasibility and functionality. There were difficulties with the proprietary nature of the machine tool controller even though it is advertised as open-architecture. The idea that custom Windows applications could be developed and installed on the machine’s Windows-based PC was attractive and proved to be useful in developing this application. However, the machine components are proprietary, so without MTConnect serving as a link between the machine and the software application the ability to access the machining process parameters for monitoring and control applications would be limited. Control applications are still limited because MTConnect is not Read-Write, and this limited the project to a monitoring application instead of a monitoring and controls application. Also because the machine tool and its controller are proprietary in nature, a monitoring system

using an external controller was required to integrate sensors that were not supported by Okuma. This required the integration between several proprietary software tools to achieve the goals of this project. Also, if the application were to be installed on the machine tool's PC, it would have required the installation of LabVIEW, the FPGA and Real-Time modules, and other associated NI software. The machine's PC has limited hard drive space and cannot accommodate these software packages, so a host PC was required for the data acquisition software and therefore to host the chatter identification application.

Integrating the proprietary NI data acquisition hardware and software components with the Visual Studio application development environment was a challenge. It would have been impossible without the collaboration of NI and Microsoft in creating the Measurement Studio toolkit for Visual Studio. This is an example of two proprietary software components that are incompatible. The Measurement Studio solution is a necessary addition to the capabilities of Visual Studio for the development of applications that wish to use NI data acquisition tools.

In addition, the learning curves are very large for using the CompactRIO, LabVIEW, and Microsoft Visual Studio, and using these tools proved to be challenging. Integrating all of these components into one system required intensive research into the product specifications, tutorials and example codes, and documentation on the various functions used by each of these sub-systems.

Long-Term Implementation Vision

This application was designed with future customizations and implementations in mind. The current version of the application was developed as a Windows form which is a standalone Windows-based application that can be deployed and installed on any PC. For a more interoperable application to be used over the Internet, the application could be developed as a “Web forms” application in Visual Studio. This would allow the application to be accessed over the Internet, but would require a server to host the application. In an industrial setting, many web applications are already hosted on web servers so this would be an easy transition for this application. However in academia, this requires a more complex setup, as web servers are not readily available over university networks to host individual web applications. A separate custom web server would need to be set up on a PC. This could be complex to program and set up for individuals with limited experience in this area, however it is possible to do if this is desired. If the application were to be hosted on the Internet ideally any device could access the application remotely, such as a smart phone, tablet, or any PC with an Internet connection. Also, a database for storing the information acquired through this application could be set up and hosted on the same server to be accessed remotely. Data security could be an issue and would need to be considered if the application and database were set up to be accessed over the Internet.

Another vision of this application was for it to be used in chatter suppression as well as identification. The force sensor could be used in tandem with an external feed rate override system that was developed by a former student at the university [49], [59] to be

used for cutting force control in turning. This system consists of a motor controlled by the CompactRIO that drives a belt system to turn the machine tool's feed rate override knob automatically based on the force setpoint defined by the user. This system could be used to automatically increase or decrease the feed rate to adjust the force input by the tool on the workpiece to help suppress chatter after it has been detected by the application. This would also require the use of the background LabVIEW program and some additional code to add the control algorithm to the program. Another method of performing automatic control of chatter conditions would be if the MTConnect standard had Read-Write capabilities, as discussed previously, in order to have access to machine parameters such as the spindle speed or the feed rate. This would allow the user to define parameters to control to suppress chatter, or the application could be written to perform the calculations automatically.

The long-term implementation vision for this application in industrial manufacturing is for it to be used as a vibration monitoring tool to detect chatter using the force sensor, but it could also be used with different sensors through the CompactRIO due to its customizable and modular nature. Also, since MTConnect can accept a wide variety of sensors, ideally, the application would be able to easily adapt to different sensors on the plots simply by dragging and dropping the sensor data items in the same way the other data items are plotted.

The application could also be useful in academia because it could be customized for individual students' experiments and would give students the ability to organize and keep track of machining process information from the machine tool that was not available

to them in post-analysis otherwise, unless they wrote that information down on paper. The organizational benefits of this application would help with their Design of Experiments (DOE) which would have multiple iterations and trials to organize. The application would be useful for students performing experiments using the force sensor, as it is already set up for use, but can be customized for use with various sensors as well as with a sensor-fusion system. If they would need to gather accurate and precise data, they would need to choose an appropriate sampling rate and further investigation of the capabilities of the force sensor and the CompactRIO would be required.

Benefits to Manufacturing Industry

The chatter identification application that was developed could benefit several different areas of machining and the manufacturing industry. The work setup abilities of the application, including the part barcode scanning and the tracking of the part information in a database, was a request from an industry partner in Manufacturing Technology at GE Power & Water. They had a need for tracking part information and providing the machine operator with part-specific machining instructions. After developing a separate application based on this request, it was clear that the application would help them improve operator efficiency and productivity in inputting part numbers and tracking this information for future analysis. The process of entering part numbers and tracking what personnel started certain jobs on specific parts would be automated by scanning personnel badges and barcodes on each part. This gives their operators more confidence that the information was correct and would allow the plant engineers the ability to organize the information more efficiently for more in-depth part-specific

analysis. Also, because of these capabilities elements of this separate application that was developed for the specific needs of GE was incorporated into the chatter identification application for this thesis and for entering the MTConnect Challenge.

The chatter identification application helps in reducing the human-machine communication gap because of the readable and interactive user interface that was developed. Many students in our research group perform machining experiments with sensors and data acquisition tools. This application will help them perform more efficient experiments because they will have all of the necessary machining process information in one central location. The ability to save this information together will help them organize their experiments and recall the data easily when performing post-processing analysis and when presenting their results.

In addition to better communication, the detection of chatter in machining could help the manufacturing industry achieve better part quality, dimensional consistency, and the ability to salvage high value raw materials that would otherwise be discarded due to the effects of chatter on the material. In areas such as the aerospace industry, it is extremely important to reduce the amount of defective and wasted raw materials and produce high quality machined components to be installed on aircraft. Installing defective parts on an aircraft could endanger the lives of the pilots, crew members, and civilians and must be avoided. Any effort to help reduce the occurrence of machining defective parts would be beneficial for this industry.

Chatter identification also leads to a reduction in the cost of replacing and re-machining parts as well as reducing tool wear and the cost of replacing tools. The amount

of machine downtime would also benefit from chatter detection. If a part is exposed to chatter and is deemed unusable, both the part and the tool will need to be replaced and therefore the machine will be down for the amount of time it takes to replace the tool and part in the machine. Even if this only takes several minutes, this is precious time that could have been put to use in machining more parts. The application will aid machine operators in identifying chatter early in the machining process instead of having to wait until the machining process was complete to check if chatter had affected the part. This makes for a more productive machining process, and therefore a more productive manufacturing industry.

The application could be used as a general monitoring tool even if the application is not used for chatter identification. The user interface was set up simply to gather sufficient machining process information in one place to monitor the machining operation in near real-time. If a manufacturing company wanted to implement this application but use different sensors to monitor different physical processes, they would be able to adapt easily based on the reconfigurable nature of the CompactRIO. The MTConnect communication process, data acquisition, and graphical displays would be identical with only the address of the MTConnect agent needed to connect to the correct machine. Therefore, the general and customizable nature of this application is beneficial to many different areas of the manufacturing industry.

Cost Benefits

This application provides several cost benefits for the manufacturing industry as well as the manufacturing research laboratories in academia. The application provides a general and intuitive user interface that operates on a familiar Windows operating system which allows any machine operator to be able to easily use the application with minimal training, reducing the cost needed to train the staff. Also, because the part tracking is done virtually there would be less of a need to provide physical recording materials such as log notebooks to keep track of part information, reducing the cost of purchasing those materials for each machine operator unless it is used as a backup recording tool.

As stated previously, the application could help reduce cost in purchasing additional raw materials and tooling to replace parts and tools that were damaged due to the effects of chatter and other destructive machining conditions by providing a way of monitoring these conditions and preventing them early in the machining operation. This also helps reduce the cost of machine downtime, which can be much more expensive than the damaged part down the line, because the application is designed to be used as a predictive maintenance tool which would help to identify these conditions before they become overly destructive. The application also helps reduce the cost of providing maintenance to the machine because if destructive machining conditions are prevented then there will be less of a need to repair the machine.

Productivity Benefits

The productivity benefits associated with this application are the following. Due to the relatively minimal training required in operating the application, the machine operator can focus on the machining process at hand instead of trying to learn how to use the software. In academia, the student performing machining experiments can focus on the experiment and its results rather than having to learn how to use and develop code in various software tools like LabVIEW which tend to have large learning curves. The students who will be using this software can gather information in a central location and will not have to worry about losing time trying to develop their own data acquisition software. Also, because all of the part information and machining process data will be saved in a central location, it can be easily accessed and is organized for productive post-processing and analysis.

The ability to identify chatter conditions using this application aid in the productivity of the machining process as well. Machine downtime would be reduced because the machining parameters could be adjusted as soon as chatter is identified which would prevent destructive conditions from occurring. The process of re-machining of defective parts would also be reduced which will increase productivity in the manufacturing environment.

Quality Benefits

The main quality benefits are due to the ability to monitor the machining process and prevent the part from being subjected to conditions that would reduce the quality and

conformity of the parts, such as chatter. This also helps to reduce the number of parts that would be discarded if they became damaged due to destructive conditions. Also, because the machining process information would be saved for each part, the ability to analyze what went wrong in the individual machining process could help improve the quality of parts in future machining operations. In addition, tracking this information would help reduce human errors in recording part information by hand and help aid in the transfer of information to downstream machining operations.

Creative and Innovative Benefits

This application is innovative in the way that it combines the open-source MTConnect standard with proprietary software and hardware into a central user-interface for data acquisition and processing. After performing research on how and where Measurement Studio can be used to integrate National Instruments products with Visual Studio, it became clear that there is currently very little documentation and implementation of this software development tool. This application introduces a way of using Measurement Studio with various data sources, either through MTConnect or through National Instruments hardware, to perform data acquisition. This is a breakthrough implementation of MTConnect due to the integration between the software development platforms to aid in solving the communication problem in manufacturing.

Feasibility of Implementation

The implementation of this application is relatively simple in operation but requires some initial setup. The implementation requires the machine tool to either be equipped with an MTConnect adapter and agent natively or the ability to have these installed. A relationship exists between Okuma machine tools and MTConnect where the adapter and agent can easily be installed on the machine if it is not already natively installed. This makes the MTConnect communication process very easy to implement in Okuma machine tools. The initial set-up procedure requires the knowledge of the machine's IP address and connecting to the adapter and running the agent after the machine start-up, but does not require any additional efforts. The external sensor setup is more complex but can run independently after the initial setup. The background LabVIEW program can be set to run at start-up on the host PC and will automatically connect to the CompactRIO to communicate the shared network variables to the NI Distributed System Manager that comes native with LabVIEW. This does require, however, that LabVIEW be installed on the host PC along with the Real-Time module. After this initial setup, the background LabVIEW program can continuously run without any interruptions. The application will call on the shared network variables to read in the external sensor data, and will also call on the MTConnect agent to gather the values of each machine's data item. As long as both of these components are running, the application can be easily implemented to monitor the machining process. Because the data communication process in this application is relatively simple, the application can

easily be used to gather the appropriate machining process data for further analysis and track the information to be used in downstream operations.

Deployment Abilities

The technologies required for this application to be integrated into a manufacturing environment are readily available for purchase and installation. The data communication methods required by this application are already in use in manufacturing facilities that have machines connected to an internal network, whether it is via Ethernet or wireless connections. The application communicates the shared network variables from the CompactRIO and LabVIEW over a network that is already in place. The inherent nature of MTConnect also communicates in this manner. This makes it easy to deploy in any manufacturing environment that communicates over these connections.

Usability

The application was developed with usability in mind. An important feature of the application is the familiar Windows platform which makes it very intuitive for the user. Once the system setup is completed, the user would simply start the application and could let it run on the machine for as much time as is needed. The addition of buttons to clear the plot and lists of data items makes it easier for the user to restart the process of monitoring a new part. The setup of the application system would require some training for those personnel involved, but the operation of the application does not require more than a few minutes to become familiar with the interface.

Implementation Costs

The application requires some costs in implementing the National Instruments products due to its proprietary nature. This would also require that the hardware such as the CompactRIO or other similar data acquisition hardware be present at each machine tool to acquire the force sensor data on each tool. For an academic research environment this is not a concern since there is usually only one or two machine tools being used, but for an industrial environment with many machine tools performing machining operations that would benefit from this application it may be costly in the beginning of this implementation. However, the benefits could outweigh the integration costs because of the potential to save many parts from the destructive nature of chatter. Because an industrial manufacturing facility usually machines hundreds if not thousands of parts each day, the cost savings would eventually add up.

Breadth of Impact

The breadth of impact of this application is wide due to the fact that the technologies of the system are readily available and easily implemented in any manufacturing process as long as the desired sensors are available for use on the machine. The application can be used to detect chatter conditions using a force sensor, but can also be customized for monitoring different machining conditions with the use of other types of sensors. This information combined with the machining parameters communicated through MTConnect would allow any manufacturing environment to use the data for whatever purpose they need. This application is more of an open solution

because of its adaptability to various machining environments. It can also be somewhat of a restricted solution due to the equipment needed to operate the external sensor data acquisition capabilities.

The quality of this application can be determined by looking at the extended functionality of the Sample Client developed by the MTConnect Institute [1] as well as the integration of the Measurement Studio tools to display the data items from MTConnect alongside the data from external sensors to help solve the communication problem in the manufacturing industry. This application is a solution to this problem that will help users keep track of part information, monitor the machining process in near-real time, and correct for any issues if destructive machining conditions occur such as chatter. This is one solution that aims to expand the usage of MTConnect as well as Measurement Studio in performing data acquisition and software development.

CHAPTER EIGHT

CONCLUSION

This thesis presented the details of the design of a custom software application to meet the needs of the manufacturing industry in communicating machining process information by using MTConnect, a data communication standard for machine tools, with sensor signals collected by proprietary data acquisition tools. The combination of using Visual Studio with NI Measurement Studio to develop the application allowed the integration between two proprietary software development tools to aid in the data acquisition process from custom sensors in the same application as data gathered through the MTConnect standard. This allowed the application to be able to be interoperable between different machine tools and other manufacturing equipment, only needing to install the MTConnect capabilities on the machine tools to be able to communicate with the application. The customizable nature of the CompactRIO allows a wide variety of sensors to be used for many different monitoring requirements. The intuitive user-interface, utilizing the familiar Windows platform in development, allows a machine operator to easily monitor the machining process remotely.

Important part-specific machining process information can be saved along with the sensor data due to the part tracking capabilities of the application. A separate application was developed for the GE Power and Water facility in Greenville, SC to aid in operator consistency and to meet their part tracking needs using a barcode scanner to track part numbers, job numbers, and personnel assigned to specific parts and jobs. A database was developed to aid in the organization of this information.

The application was designed to monitor and detect chatter conditions using a force sensor and machining process information through MTConnect to provide the machine operator with information to adjust the appropriate parameters to reduce and suppress chatter during operation and prevent chatter in future operations. A simulation was performed to detect vibrations through the force sensor using a vibration shaker table which would represent a chatter condition on the machine tool. This simulation was integrated with a machining simulator at the Okuma North America facility to connect to the MTConnect agent and view the machining parameters from a simulation of a cutting process on a lathe. This showed that the chatter identification application could successfully sense variations in force attributed to vibrations in the machining process, also showing the capabilities of performing a spectral analysis through Measurement Studio tools, and directly relate chatter occurrence to machining parameters through MTConnect.

Other implementations could be utilized through this application due to its customization abilities and the interoperable nature of the application using the MTConnect standard. Other students can benefit from using the application in their experiments to gather information that would normally have been “locked” in the machine controller and track their design of experiments in an organized fashion. Similarly, the application can be used as a general monitoring tool and can be customized for a wide variety of uses to meet the needs of the manufacturing industry.

This application was attractive to the judges of the MTConnect Challenge, which is supported by the U.S. Department of Defense, and allowed this application to advance to the top 5 applications to be considered in the final round of judging. This will take place at the MC2 Conference on April 8-10, 2014, consisting of a live demonstration of the chatter simulation capabilities along with the use of Okuma's machine simulator to gather the machining process parameters through the MTConnect agent on the simulator.

Overall, this application meets the objectives defined for the project to design a customizable software application utilizing the MTConnect standard and custom sensors to monitor and communicate machining process information for the means of identifying chatter conditions. The project outlined in this thesis provides a solution to help meet the needs of the manufacturing industry for an interoperable application utilizing a standard form of data communication.

REFERENCES

- [1] Schmitz, Tony L., and Kevin S. Smith. *Machining Dynamics: Frequency Response to Improved Productivity*. New York: Springer, 2008. Print.
- [2] "Machining Navi." *Okuma.com*. Okuma The Americas, n.d. Web. 22 Apr. 2014. <<http://www.okuma.com/machining-navi>>.
- [3] "The MTConnect Institute." *The MTConnect Institute*. N.p., n.d. Web. 24 Oct. 2013. <<http://www.mtconnect.org/>>.
- [4] "Introduction to XML." W3Schools.com. W3Schools, n.d. Web. 22 Apr. 2014. <http://www.w3schools.com/xml/xml_what_is.asp>.
- [5] "What, How & Why." MTConnect.org. MTConnect Institute, n.d. Web. 22 Apr. 2014. <<http://mtconnect.org/overview/mtconnect-overview/what%2C-how-why.aspx>>.
- [6] "Getting Started with MTConnect: Architecture." Mtconnect.org. MTConnect Institute, 5 Sept. 2012. Web. 1 Feb. 2014. <http://mtconnect.org/media/33639/mtconnectarchitecture_2012-10-26.pdf>.
- [7] Getting Started with MTConnect. Tech. AMT- The Association for Manufacturing Technology, 1 Oct. 2011. Web. 22 Apr. 2014. <http://mtconnect.org/media/7312/getting_started_with_mtconnect_-_final.pdf>.
- [8] "RESTful Web Services: The Basics." DeveloperWorks. IBM, n.d. Web. 22 Apr. 2014. <<https://www.ibm.com/developerworks/webservices/library/ws-restful/>>.
- [9] "MTConnect Challenge 1." Challenge Post. N.p., 2013. Web. 02 Apr. 2014. <<http://mtconnect.challengepost.com/>>.
- [10] "MTConnect Challenge 2." Challenge Post. MTConnect Institute, 2013. Web. 02 Apr. 2014. <<http://mtconnect2.challengepost.com/>>.
- [11] "[MC2] 2014 Conference." [MC2] MTConnect: Connecting Manufacturing Conference. MTConnect Institute, n.d. Web. 22 Apr. 2014. <<http://mtconnectconference.org/>>.
- [12] Merchant, M. E., Dornfeld, D. A., and Wright, P., K. (2005), "Manufacturing — Its Evolution and Future", Trans. North American Manufacturing Research Institute, vol. 33, pp. 211-218

- [13] Pugazhendi, M., and R. Krishnamurthy. "Condition Monitoring of Drill Process Using a Telemetry System." *Journal of Materials Processing Technology* 28.1-2 (1991): 67-73. Web. 22 Apr. 2014.
- [14] Hosmon, John H., and Dan Fritz. "Connecting Your Legacy CNC Machine Tool To The Internet." *Modern Machine Shop*. Modern Machine Shop, 1 May 2003. Web. 22 Apr. 2014. <<http://www.mmsonline.com/articles/connecting-your-legacy-cnc-machine-tool-to-the-internet>>.
- [15] Milfelner, M., F. Cus, and J. Balic. "An Overview of Data Acquisition System for Cutting Force Measuring and Optimization in Milling." *Journal of Materials Processing Technology* 164-165 (2005): 1281-288. Web. 22 Apr. 2014.
- [16] Xu, X.w., and S.t. Newman. "Making CNC Machine Tools More Open, Interoperable and Intelligent—a Review of the Technologies." *Computers in Industry* 57.2 (2006): 141-52. Web. 22 Apr. 2014.
- [17] Atluru, S., A. Deshpande, S. Huang, and J. P. Snyder. "Smart Machine Supervisory System: Concept, Definition and Application." *Society for Machinery Failure Prevention Technology 62nd Conference*. Proc. of Society for Machinery Failure Prevention Technology 62nd Conference. 6-8. Google Scholar. Web. <<http://scholar.google.com/citations?user=XICvOyQAAAAJ&hl=en>>.
- [18] Vijayaraghavan, Athulan, Will Sobel, Armando Fox, David Dornfeld, and Paul Warndorf. "Improving Machine Tool Interoperability Using Standardized Interface Protocols: MT Connect." *Green Manufacturing and Sustainable Manufacturing Partnership, Laboratory for Manufacturing and Sustainability* (2008): n. pag. EScholarship University of California. Web. 2 Apr. 2014.
- [19] Teti, R., K. Jemielniak, G. O'Donnell, and D. Dornfeld. "Advanced Monitoring of Machining Operations." *CIRP Annals - Manufacturing Technology* 59.2 (2010): 717-39. Web. 22 Apr. 2014.
- [20] Ramesh, R., S. Jyothirmai, and K. Lavanya. "Intelligent Automation of Design and Manufacturing in Machine Tools Using an Open Architecture Motion Controller." *Journal of Manufacturing Systems* 32.1 (2013): 248-59. Web. 22 Apr. 2014.
- [21] Vijayaraghavan, Athulan,, and David Dornfeld. "Addressing Process Planning and Verification Issues Using MTConnect." *Precision Manufacturing Group, Laboratory for Manufacturing and Sustainability* (2009): n. pag. EScholarship University of California. Web. 2 Apr. 2014

- [22] Vijayaraghavan, A., and D. Dornfeld. "Automated Energy Monitoring of Machine Tools." *CIRP Annals - Manufacturing Technology* 59.1 (2010): 21-24. Science Direct. Web. 24 Oct. 2013.
<<http://www.sciencedirect.com/science/article/pii/S0007850610000430>>.
- [23] Cheng, Hsin-Yu, Yu-Chuan Su, and Jo-Peng Tsia. "Design of a Machine Tool Information Service System." *2010 International Symposium on Computer Communication Control and Automation (3CA) 2* (2010): 33-36. Web. 2 Apr. 2014.
- [24] Kadir, Aini Zuhra Abdul, Xun Xu, and Firman Ridwan. "Employing Actual Shop-floor Status and Machine Tool Kinematic Data Model in Machining Simulation." *International Journal of Integrated Engineering* 3.1 (2011): n. pag. Web. 2 Apr. 2014.
- [25] Malik, Subhasish, and Mark D. Bedillion. "Event-Based Temperature Control for Machining Using MTConnect." *Dynamics, Control and Uncertainty, Parts A and B. Proc. of ASME 2012 International Mechanical Engineering Congress and Exposition, Houston, Texas, USA. Vol. 4. N.p.: ASME, 2012. N. pag. ASME Digital Collection.* Web. 2 Apr. 2014.
- [26] Atluru, Sri H., and Amit Deshpande. "Data to Information: Can MTConnect Deliver the Promise?" *Transactions of NAMRI/SME* 37 (2009): 197-204. Web. 2 Apr. 2014.
- [27] Atluru, Sri H., Amit Deshpande, Sam Huang, and Ron Pieper. *ASME 2012 International Manufacturing Science and Engineering Conference (MSEC2012)* (2012): n. pag. Web. 2 Apr. 2014.
- [28] Michaloskia, John L., Y. F. Zhaob, B. E. Leea, and W. G. Rippeya. "Web-enabled, Real-time, Quality Assurance for Machining Production Systems." *The Twelfth CIRP Conference on Computer Aided Tolerancing. Vol. 10. N.p.: CIRP, 2010. 332-39.* Science Direct. Web. 2 Apr. 2014.
- [29] "Official MTConnect Demo." *Official MTConnect Demo. MTConnect Institute, n.d.* Web. 22 Apr. 2014. <<http://imts.mtconnect.org/>>.
- [30] Lacey, S. J. "The Role of Vibration Monitoring in Predictive Maintenance Part 1- Principles and Practice." *Maintenance & Asset Management. Maintenance & Engineering Magazine, Jan.-Feb. 2010.* Web. 22 Apr. 2014.
<http://maintenanceonline.org/maintenanceonline/content_images/p36-45_%20Lacey%20Paper%201.pdf>.

- [31] Lacey, S. J. "The Role of Vibration Monitoring in Predictive Maintenance Part 2- Some Illustrative Examples" *Maintenance & Asset Management. Maintenance & Engineering Magazine*, Mar.-Apr. 2010. Web. 22 Apr. 2014.
<http://maintenanceonline.org/maintenanceonline/content_images/Page%2038,39,40,41,42,43,44.pdf>.
- [32] Lacey, S. J. "The Role of Vibration Monitoring in Predictive Maintenance." INA FAG. Schaeffler Group Industrial, n.d. Web. 22 Apr. 2014.
<<http://source.theengineer.co.uk/Journals/2012/09/06/b/v/m/The-role-of-vibration-monitoring-in-predictive-maintenance.pdf>>.
- [33] Gegg, Brandon C., C. Steve. Suh, and Albert C. J. Luo. *Machine Tool Vibrations and Cutting Dynamics*. New York: Springer, 2011. Web.
- [34] Wu, D. W. "Comprehensive Dynamic Cutting Force Model and Its Application to Wave-Removing Process." *J. Manuf. Sci. Eng* 110.2 (1988): 153-61. ASME Digital Collection. ASME, 30 July 2009. Web. 22 Apr. 2014.
- [35] Zhang, Wen-Guo, Jun-Yi Yu, Si-Mao Qiao, and Fu-Lun Yang. "A New Approach to the Early Prediction of Turning Chatter." *Journal of Vibration and Acoustics* 116 (1994): 485-88. ASME Digital Collection. ASME, 17 June 2008. Web. 22 Apr. 2014.
- [36] Eman, K. F., and S. M. Wu. *Forecasting Control of Machining Chatter*. Tech. University of Wisconsin-Madison, n.d. Web. 22 Apr. 2014.
<<http://www.mech.northwestern.edu/MFG/AML/pdf/1980-2.pdf>>.
- [37] Eman, K. F. *Identification and Control of Chatter in Turning*. Tech. University of Wisconsin-Madison, n.d. Web. 22 Apr. 2014.
<<http://www.mech.northwestern.edu/MFG/AML/pdf/1984-17.pdf>>.
- [38] Han, Xianguo, Huajiang Ouyang, Minjie Wang, Nurhafizzah Hassan, and Yuming Mao. "Self-excited Vibration of Workpieces in a Turning Process." *Journal of Mechanical Engineering Science* 226.8 (2012): 1958-970. Sage Journals. Sage Publications, 19 Jan. 2012. Web. 22 Apr. 2014.
<<http://pic.sagepub.com/content/226/8/1958>>.
- [39] Schmitz, Tony L. *Stability Diagrams for High-speed Milling*. PowerPoint Presentation. University of Florida, n.d. Web. 22 Apr. 2014.
- [40] Wang, Min, and Renyuan Fei. "On-line Chatter Detection and Control in Boring Based on an Electrorheological Fluid." *Mechatronics* 11.7 (2001): 779-92. Web. 22 Apr. 2014.

- [41] Mei, Deqing, Tianrong Kong, Albert J. Shih, and Zichen Chen. "Magnetorheological Fluid-controlled Boring Bar for Chatter Suppression." *Journal of Materials Processing Technology* 209.4 (2009): 1861-870. Web. 22 Apr. 2014.
- [42] Sajedipour, D., S. Behbahani, and S. M.K. Tabatabaei. "Mechatronic Modeling and Control of a Lathe Machine Equipped with a MR Damper for Chatter Suppression." 2010 8th IEEE International Conference on Control and Automation. Xiamen, China: IEEE, 2010. 802-07. Web.
- [43] Al-Regib, Emad, Jun Ni, and Soo-Hun Lee. "Programming Spindle Speed Variation for Machine Tool Chatter Suppression." *International Journal of Machine Tools and Manufacture* 43.12 (2003): 1229-240. Web. 22 Apr. 2014.
- [44] Moradi, Hamed, M.r. Movahhedy, and G. Reza Vossoughi. "Robust Control Strategy for Suppression of Regenerative Chatter in Turning." *Journal of Manufacturing Processes* 11.2 (2009): 55-65. Web. 22 Apr. 2014.
- [45] Moradi, H., M. R. Movahhedy, and G. Vossoughi. "Sliding Mode Control of Machining Chatter in the Presence of Tool Wear and Parametric Uncertainties." *Journal of Vibration and Control* 16.2 (2010): 231-51. Web. 22 Apr. 2014.
- [46] Mehta, Parikshit, and Laine Mears. "Model Based Prediction and Control of Machining Deflection Error in Turning Slender Bars." *Proceedings of the Sixth Annual ASME International Manufacturing Science & Engineering Conference*. Corvallis, Oregon. N.p.: ASME, 2011. N. pag. Print.
- [47] Mehta, Parikshit, Matthew Kuttolamadom, and Laine Mears. "Machining Process Monitoring: Bayesian Update of Machining Power Model." *Proceedings of the Seventh Annual ASME International Manufacturing Science & Engineering Conference*. Notre Dame, Indiana. N.p.: ASME, 2012. N. pag. Print.
- [48] "About the Manufacturing Sector." *Industries at a Glance*. U.S. Bureau of Labor Statistics, 18 Apr. 2014. Web. 22 Apr. 2014. <<http://www.bls.gov/iag/tgs/iag31-33.htm#about>>.
- [49] Mehta, Parikshit, and Laine Mears. "Cutting Force Control in Machining: Bayesian Update of Mechanistic Force Model." *ASME 2012 5th Annual Dynamic Systems and Control Conference Joint with the JSME 2012 11th Motion and Vibration Conference*. Fort Lauderdale, FL. N.p.: ASME, 2012. N. pag. Print.
- [50] Sosnicki, Olivier, A. Pages, Carinne Pacheco, and Thomas Maillard. "Servo Piezo Tool SPT400MML for the Fast and Precise Machining of Free Forms." *The International Journal of Advanced Manufacturing Technology* 47.9-12 (2010): 903-10. Web. 22 Apr. 2014.

- [51] "Timeline of Computer History : Computers Entries." Exhibits. Computer History Museum, 2006. Web. 21 Apr. 2014.
<<http://www.computerhistory.org/timeline/?category=cmptr>>.
- [52] Flom, D. G., R. Komanduri, and M. Lee. "High-Speed Machining of Metals." Annual Review of Materials Science 14.1 (1984): 231-78. Web. 22 Apr. 2014.
- [53] "Okuma's PREX Motor Wins Superior Energy-Saving Machine President's Award from Japan Machinery Federation." Newsroom 2010. Okuma Corporation, 25 Feb. 2010. Web. 22 Apr. 2014. <<http://www.okuma.co.jp/english/news/others/100225.html>>.
- [54] Ditton, Thames. "'Secret' Motor Drives Lathe Spindles as Well as Turrets and Live Tools." Manufacturing Engineering News. IndustrialPR, n.d. Web. 22 Apr. 2014.
<<http://industrialpr.net/%25E2%2580%2598secret%25E2%2580%2599-motor-drives-lathe-spindles-as-well-as-turrets-and-live-tools/>>.
- [55] "LB4000 EX." CNC Machine. Okuma - The Americas, n.d. Web. 1 Feb. 2014.
<<http://www.okuma.com/lb4000-ex>>.
- [56] "THINC-OSP." CNC Control; Control Technology; OSP. Okuma The Americas, n.d. Web. 22 Apr. 2014. <<http://www.okuma.com/thinc-osp>>.
- [57] "Advantages of NI CompactRIO." NI.com. National Instruments, 15 Jan. 2009. Web. 02 Apr. 2014. <<http://www.ni.com/white-paper/8151/en/>>.
- [58] "Type K." Revised Thermocouple Reference Tables. Omega, n.d. Web. 22 Apr. 2014. <<http://www.omega.com/temperature/Z/pdf/z204-206.pdf>>.
- [59] Mehta, Parikshit, and Laine Mears, PhD, PE. Machining Force Sensor Report. Tech. N.p.: n.p., n.d. Print.
- [60] "NI CompactRIO." NI.com. National Instruments, n.d. Web. 22 Apr. 2014.
<<http://www.ni.com/compactrio/>>.
- [61] "NI CRIO-9022." NI.com. National Instruments, n.d. Web. 02 Apr. 2014.
<<http://sine.ni.com/nips/cds/view/p/lang/en/nid/206760>>
- [62] "NI 9215." NI.com. National Instruments, n.d. Web. 02 Apr. 2014.
<<http://sine.ni.com/nips/cds/view/p/lang/en/nid/208793>>.
- [63] "NI 9211." NI.com. National Instruments, n.d. Web. 02 Apr. 2014.
<<http://sine.ni.com/nips/cds/view/p/lang/en/nid/208787>>.

- [64] "LabVIEW System Design Software." NI LabVIEW. National Instruments, n.d. Web. 22 Apr. 2014. <<http://www.ni.com/labview/>>.
- [65] Tacke, Chris. "Building a Client." Building a Client. MTConnect Institute, n.d. Web. 31 Jan. 2014. <<http://mtconnect.org/getting-started/tutorials/building-a-client.aspx>>.
- [66] "MTConnect Managed SDK." CodePlex. N.p., n.d. Web. 22 Apr. 2014. <<http://mtconnect.codeplex.com/>>.
- [67] "NI CRIO-9022." National Instruments, 14 Apr. 2011. Web. 22 Apr. 2014. <<http://sine.ni.com/ds/app/doc/p/id/ds-202/lang/en>>.
- [68] "SR61 Rugged 1D Handheld Scanner." Intermec by Honeywell, n.d. Web. 22 Apr. 2014. <<http://www.intermec.com/products/scansr61/index.aspx>>.
- [69] Weaver, Jon. "Using .NET Interfaces for Machine Neutral THiNC Applications." Thincster - The Unofficial Okuma THiNC API Solution Center. Thincster, 16 Feb. 2012. Web. 22 Apr. 2014. <<http://thincster.com/2012/02/16/using-net-interfaces-for-machine-neutral-thinc-applications/>>.
- [70] Weaver, Jon. "An Introduction to Developing with the Okuma THiNC API." Thincster - The Unofficial Okuma THiNC API Solution Center. Thincster, 16 Feb. 2012. Web. 22 Apr. 2014. <<http://thincster.com/2012/02/16/an-introduction-to-developing-with-the-okuma-thinc-api/>>.
- [71] "Introduction to SQL." W3Schools.com. W3Schools, n.d. Web. 22 Apr. 2014. <http://www.w3schools.com/sql/sql_intro.asp>.
- [72] Sobel, William. MTConnect Standard Part 1 - Overview and Protocol. Publication. MTConnect Institute, 17 Feb. 2012. Web. 22 Apr. 2014. <http://mtconnect.org/media/23617/mtc_part_1_overview_v1.2.pdf>.
- [73] Turner, John. MTConnect Standard Part 2 - Components and Data Items. Publication. MTConnect Institute, 17 Feb. 2012. Web. 22 Apr. 2014. <http://mtconnect.org/media/23617/mtc_part_1_overview_v1.2.pdf>.
- [74] Turner, John. MTConnect Standard Part 3 - Streams, Events, Samples, and Conditions. Publication. MTConnect Institute, 16 Jul. 2012. Web. 22 Apr. 2014. <http://mtconnect.org/media/23617/mtc_part_1_overview_v1.2.pdf>.
- [75] Sobel, William. MTConnect Standard Part 4 - Assets. Publication. MTConnect Institute, 17 Feb. 2012. Web. 22 Apr. 2014. <http://mtconnect.org/media/23617/mtc_part_1_overview_v1.2.pdf>.

- [76] Zhao, Fiona, and Frederick M. Proctor. "Publication Citation: A Roadmap for STEP-NC Enabled Interoperable Manufacturing." NIST Manuscript Publication Search. National Institute of Standards and Technology (NIST), 17 June 2011. Web. 22 Apr. 2014. <http://www.nist.gov/manuscript-publication-search.cfm?pub_id=908419>.
- [77] Proctor, Fred, Bill Rippey, John Horst, Joe Falco, and Tom Kramer. Interoperability Testing for Shop Floor Measurement. Rep. National Institute of Standards and Technology (NIST), n.d. Web. 22 Apr. 2014. <http://www.nist.gov/customcf/get_pdf.cfm?pub_id=823620>.
- [78] "Manufacturing Modeling and Simulation." Manufacturing Modeling and Simulation. National Institute of Standards and Technology (NIST), 1 Oct. 2011. Web. 22 Apr. 2014. <<http://www.nist.gov/el/isd/cs/manmodsim.cfm>>.
- [79] "NIST and Partners Offer Solution to Communications Impasse in Factories." Engineering Laboratory. NIST Tech Beat, 28 May 2013. Web. 22 Apr. 2014. <<http://www.nist.gov/el/isd/robot-052813.cfm>>.
- [80] Lee, Yung-Tsun Tina, Ju Yeon Lee, Frank Riddick, Don Libes, and Deogratias Kibira. "Interoperability for Virtual Manufacturing Systems." Int. J. Internet Manufacturing and Services 3.2 (2013): 99-120. NIST. Web. 22 Apr. 2014. <http://www.nist.gov/customcf/get_pdf.cfm?pub_id=912982>.
- [81] "Vibrators." Buffalo Dental. Buffalo Dental Manufacturing Co., Inc., n.d. Web. 2 Apr. 2014. <<http://www.buffalodental.com/pdfnew/vibrator.pdf>>.
- [82] Zhao, Y. F., T. Kramer, W. Rippey, J. Horst, and F. Proctor. "An Integrated Data Model for Quality Information Exchange in Manufacturing Systems." Proc. of 37th MATADOR Conference. National Institute of Standards and Technology (NIST), n.d. Web. 22 Apr. 2014. <http://www.nist.gov/customcf/get_pdf.cfm?pub_id=910781>.
- [83] Merchant, M. Eugene. "Production: A Dynamic Challenge: What's Needed Is Flexibility: The Ability to Manufacture Efficiently a Constantly Changing Variety of Products. The Computer Makes It Possible." IEEE Spectrum 20.5 (1983): 36-39. Print.
- [84] Stein, J., and D. A. Dornfeld. "Integrated Design and Manufacturing for Precision Mechanical Components." Proc. 1st Int'l. Conf. on Integrated Design and Manufacturing. Nantes, France: PRIMECA, 1996. 195-204. Print.
- [85] *Technical Manual Transducer Amplifier Type DR7DC*. Tech. RDP Group, n.d. Web. 22 Apr. 2014. <<http://www.rdpe.com/cds/cd2401.pdf>>.