2019-06-01

The Conversion of Manual Machining Equipment into Smart, Connected Systems with Real-Time Monitoring and Issue Identification Capabilities

David Lee Williams
Brigham Young University

The Conversion of Manual Machining Equipment into Smart, Connected Systems with Real-Time Monitoring and Issue Identification Capabilities

David Lee Williams

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Yuri Hovanski, Chair
Reuben D. Domike
Michael P. Miles

School of Technology

Brigham Young University

ABSTRACT

The Conversion of Manual Machining Equipment into Smart, Connected Systems with Real-time Monitoring and Issue Identification Capabilities

David Lee Williams
School of Technology, BYU
Master of Science

With the advent of the fourth industrial revolution, information technology and manufacturing systems are merging to form what is now known as Smart Manufacturing. However, with this newer technology being integrated with newer pieces of machining equipment, companies with legacy equipment occasionally are in a bind since these machines were not designed or built with the fundamental components of smart manufacturing systems: unified connectivity, real-time monitoring, and issue identification.

The purpose of this research is to provide a solution for converting manual machining equipment into smart systems with these fundamental components of smart manufacturing. The pieces of equipment that were the subjects of this experimentation were an HJ-1100 Kingston lathe and four ACER Vertical Turret Milling machines. None of these machines had any of these capabilities at the inception of this project.

These machines were successfully converted into smart systems with varying degrees of reliability between the lathe and the four mills in the case of real-time monitoring and issue identification. The setups and configurations to achieve these three smart components are described and provided.

Keywords: smart manufacturing, thingworx, kepserverex, unified connectivity, real-time monitoring, issue identification, lathe, mill, manufacturing apps

ACKNOWLEDGEMENTS

I wish to express appreciation to everyone who contributed towards the completion of this project, I could not have done this without them. Dr. Yuri Hovanski provided superb mentorship, insight, and guidance towards the conversion of manual machines into smart systems. Kevin Cole helped tremendously with the National Instruments side of the smart factory, especially when it came to creating and customizing LabVIEW code. John Reinhard aided significantly in the IT side of the project, helping establish connectivity with the machines and installing key software components. Clint Bybee was also instrumental in bringing machining knowledge, experience, and expertise to this project and was invaluable when simulating safe crash tests on the equipment. PTC also helped tremendously with the donation of several pieces of software, including ThingWorx and KEPServerEX.

I also express my gratitude to those that may have not been mentioned, but may have aided me in other ways, helping me get through the ups and downs, the good days and bad ones. This includes members of my family, friends, colleagues, company representatives who helped in support cases, and God. Words do not sufficiently express how much all of this support has meant to me. Thank you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

1.1 Background

As society becomes more and more immersed in the Information Age, the ability to collect, transfer, store, analyze, and optimize data becomes essential. Information technology has become one of the driving factors in innovation. This is becoming more prominently seen with the emergence of smart, connected products in essentially all manufacturing sectors, and even extends to the manufacturing processes themselves (Porter and Heppelmann, 2014). Data is being used to better optimize manufacturing assembly lines and inform line decisions. Machines now need to be able to connect and communicate with each other and provide feedback to the operator. They need to be smarter.

This gives companies three options when deciding whether or not to implement a smart solution: do nothing, purchase machines and equipment that come prebuilt with these smart capabilities, or alter their current equipment to have that connectivity capability (Muhuri, Shukla, and Abraham, 2019). The following scenario describes what typically occurs when encountered with such a decision.

In a case study shared by PTC, 3D Systems, like most manufacturers, had a system that was designed for connectivity and creating data, but no efficient way to make use of, or analyze that data. Eventually, 3D Systems was able to make use of that connectivity, and subsequently

enable their machines to have smart capabilities. These changes allowed them to start tracking 3D printer properties such as nitrogen levels and temperature variances. Furthermore, they were able to remotely diagnose and service the printers if needed (PTC, 2018). 3D Systems has seen vast improvements in its ability to collect pertinent information and resolve customer issues with greater speed and efficiency since converting their processes into smart systems.

This case study provides useful insight into both what is being done and what still needs to be explored in smart implementation. In this study, it mentions machines that already had the ability to connect. What about the machines that were never designed to do so? In exploring smart implementation, many case studies, like the one above, often involve the use of newer technology to implement a smart solution, but there is a gap in solutions for older, manual machines, such as mills and lathes. Finding a way to transform these manual machines into smart systems provides an avenue for exploration and has the potential to yield performance results similar to those of 3D Systems in the manufacturing sector.

1.2 Purpose of Research

The purpose of this research is to demonstrate that manual machining equipment can be converted into systems with the fundamental smart components of unified connectivity, real-time monitoring, and issue identification capabilities.

1.3 Research Questions

This study aims to show that manual lathes and manual mills, although never initially designed to provide information or feedback, can be converted into smart connected systems

with unified connectivity, real-time monitoring, and issue identification capabilities. This research intends to answer the following questions:

1. Is it possible to establish unified connectivity between manual lathes/mills and a computer?
2. Can the information being received from these machines be reliably manipulated in order to display real-time monitoring?
3. From the data being transferred and monitored, can issue identification be implemented to alert shop supervisors in the event of anomalies?

1.4 Methodology

One manual lathe and four manual mills were used as the subjects of experimentation. The reasoning for selecting lathes and mills is because they are two of the most common manual material removal machines in the manufacturing industry, past and present. These machines would be equipped with the hardware necessary to allow them to create and transfer data to one central computer, thus establishing unified connectivity.

That data would then be monitored on a PTC IIoT platform called ThingWorx Composer and displayed using the Thingworx Manufacturing Apps, accomplishing the second objective, real-time monitoring. The reason that PTC was chosen among other IoT platforms is because it is the leading IIoT platform in the manufacturing industry (McAvoy, 2019). ThingWorx also has the ability to establish quick, easy, and reliable connections with KEPServerEX, a piece of software intended to handle communication with the machines. KEPServerEX was used because of its ability to communicate with a wide range of protocols and machine languages, greatly simplifying the task of bringing disparate systems into one streamlined data stream that

integrates with ThingWorx Composer, which then displays the desired information (in real-time) in a visualization tool called a mashup. Combining KEPServerEX with ThingWorx presented a robust solution with a wide range of potential applications for others to utilize (Kepware, 2019).

In addition to establishing unified connectivity with these machines and having real-time, useful data being monitored, logic would then be written and events/alerts would then be constructed that analyzed that data and discerned various machine states and notified the shop supervisor of machine state anomalies, such as abnormal hours of machine operation, and unsafe RPM rates. This automated response to analyzed internal data is known as issue identification.

At the outset of this research, coordination was made with Clint, the shop supervisor, and Eric, the department tooling/machine specialist. Table 1-1 states the information that they desired to obtain from the machines being used in this research and in future research projects.

Table 1-1: Desired Information in the Smart Factory

Desired Data		
Data	**Notes on Implementation**	**How This is Going to be Accomplished**
1. Machine/spindle-spinning or stopped	Is the motor running?	That is a voltage measurement
2. Spindle RPM	Similar to a bike tire sensor for RPM, reflect the rpm of the machine	Research sensors that can connect to LabVIEW
3. Spindle load	Find out more: will this require a force sensor?	We may/may not be able to measure this
4. Crash notification	Find out more: is it a result of a dramatic change in rpm, torque, etc.?	conditional statement based on change in rpm
5. How many hours the spindle is in operation	Time stamp once data changes from 0 and then back to 0	conditional statement based on the voltage change from off/on

The methodology and results chapters in this paper will explain and summarize which of these aspects were able to be achieved and which, if any, were not able to be realized by the end of this research.

There are multiple tiers of smart manufacturing. To reach the top tier with closed-loop improvement (machine self-correction and optimization) and performance benchmarking is unrealistic in this stage of smart implementation and would take more time and resources than presently allotted. The fundamentals must first be achieved before considering more sophisticated tools and features. Furthermore, because the subjects of experimentation are located in an educational environment, features such as reverse control and self-correction will not be explored because these features are not conducive in this setting, for safety and liability reasons. It is important to note that this research and methodology is being conducted in an educational machining lab: there will be inherent differences in the data that is collected, and the features that will be implemented. There will be many similarities, but a few differences. One main example of this distinction will be demonstrating reverse control of the machines. Because this lab is set up to introduce students to using and operating these machines, reverse control of the machines will not be implemented, but in an industrial factory setting, this may be attempted and even encouraged. Another facet that is an important distinction machines responding to the data of other machines. In this setting, that will not be appropriate, but in the field, this would be an excellent way to optimize line efficiency and increase line automation.

For this education setting, the pieces of data and overall smart solution implementation will have the main goals to improve safety, increase proper education of using the different machines, and encourage correct machine usage. These goals are shared in the industrial sector, but they are not the primary goals like they are in the educational setting.

Future projects may include integrating augmented reality (AR) experiences for training and maintenance purposes, predictive analytics, and other smart principles, but the primary scope of this research was concerned with providing these manual machines with the three fundamental smart components of 1) unified connectivity, 2) real-time monitoring, and 3) issue identification. These components will be discussed more in-depth in the following chapters.

1.5 Glossary

IoT – internet of things

AR – augmented reality

ThingWorx Composer – IoT platform used to create digital twins of real-world, physical objects. This is the IoT platform used for this project

ThingWorx Manufacturing Apps – an extension of ThingWorx Composer containing prebuilt mashups, real-time monitoring, and alert capabilities with manufacturing facilities as the intended user

KEPServerEX – a connectivity platform that enables users to connect, manage, monitor, and control diverse automation devices and software applications through a single-server interface

RPM – revolutions per minute

cDAQ – compact data acquisition device. This is the type of instrument from National Instruments that was used in the lathe portion of this project

LabVIEW – software that controls the cDAQ and manipulates its data

OPC UA – Open Platform Communication Unified Architecture. The communication protocol used to connect LabVIEW to KEPServerEX

SMS – smart manufacturing system

2 LITERATURE REVIEW

2.1 Introduction

It has been suggested that up to this point in history, there have been four defining industrial revolutions. The first was the advent of steam power systems. The second began with the introduction of electricity and assembly line mass production. The third came with the arrival of computers and information technology. The fourth industrial revolution, the one in which we are now participating, involves machine communication and increased levels of automated functions (Muhuri, Shukla, and Abraham, 2019). Data creation and analysis is now the driving factor for decision making. Smart manufacturing is becoming the focus of global manufacturing transformation and is changing the way that companies and industries operate (Qi and Tao, 2018).

This is evident in an initiative performed by the National Institute of Standards of Technology (NIST) in an endeavor to have a 22 kW lathe become more “smart”. Examples of the pieces of data that NIST was able to track and monitor include spindle load, spindle speed, cutting, thrust, and axial forces. They demonstrated that by creating data and analyzing that data, optimization and first part correct production can be achieved (Ivester and Heigel, 2000). This versatility of production leads to dramatic increases in levels of innovation and commensurate decreases in changeover time. From this case, some of the possibilities are shown for what can be done to a lathe machining station, and provides a good example for some of the features that

will be integrated into the lathe and mills serving as the subjects for this research. Their CNC lathe was able to generate and send data, manipulate it into a usable format, and respond to it, which will be the goal for this manual lathe. The difference is the CNC lathe came with those capabilities prebuilt, and the manual lathe and mills did not. By providing a solution to allow manual machines to have these capabilities, companies have another avenue to use to maintain a competitive edge.

As shown in the NIST example, data is powerful. It helps companies reduce material and machining costs, optimize line operations, and provide supervisors with useful data for market strategy and growth. With more and more devices becoming connected, the possibilities and progress that can and will be made will accelerate. In 2015, there were approximately 15.41 billion devices connected. This year, in 2018, there are approximately 23.14 billion devices, and in 2025, it is expected that there will be above 75 billion connected devices (Statista, 2018). More and more companies are starting to embrace this relatively new concept, converting their "dumb" machines into "smart" systems.

2.2 Defining "Smart"

The term "smart" can have multiple interpretations and meanings. In general, people think of a smart phone, or a smart TV: a device that is able to able to connect, share, and interact with a person or other device (Techopedia, 2019). But what does the word "smart" connote in the industrial sector? "Smart" when applied to an industrial system refers to fully-integrated, collaborative systems that respond in real time to meet changes in the factory setting, the supply network, and customer needs (McKewen, 2015). It is associated with 4 main capabilities: self-recognition and communication to other parts of the manufacturing enterprise, self-monitoring

and optimization of its operations, self-assessment of the quality of its work, and self-learning and performance improvement over time (Ivester and Heigel, 2000). The word "self" is associated with all of these capabilities, implying varying levels of machine independence in smart systems. In this research project, the first three of these four capabilities were implemented. The fourth level of self-learning extends beyond the scope of the environment in which these machines are stationed. The first three capabilities of communication, self-monitoring, and self-assessment will be referred to as unified connectivity, real-time monitoring, and issue identification respectively throughout the rest of this paper, and will be discussed in greater detail. These components constitute what PTC calls the understand phase of the manufacturing digital transformation journey (Figure 2-1). This project will cover the connectivity, visibility, and issue identification portions in the understand phase of the journey, but has the potential with further experimentation to venture into the advance and outperform phases.

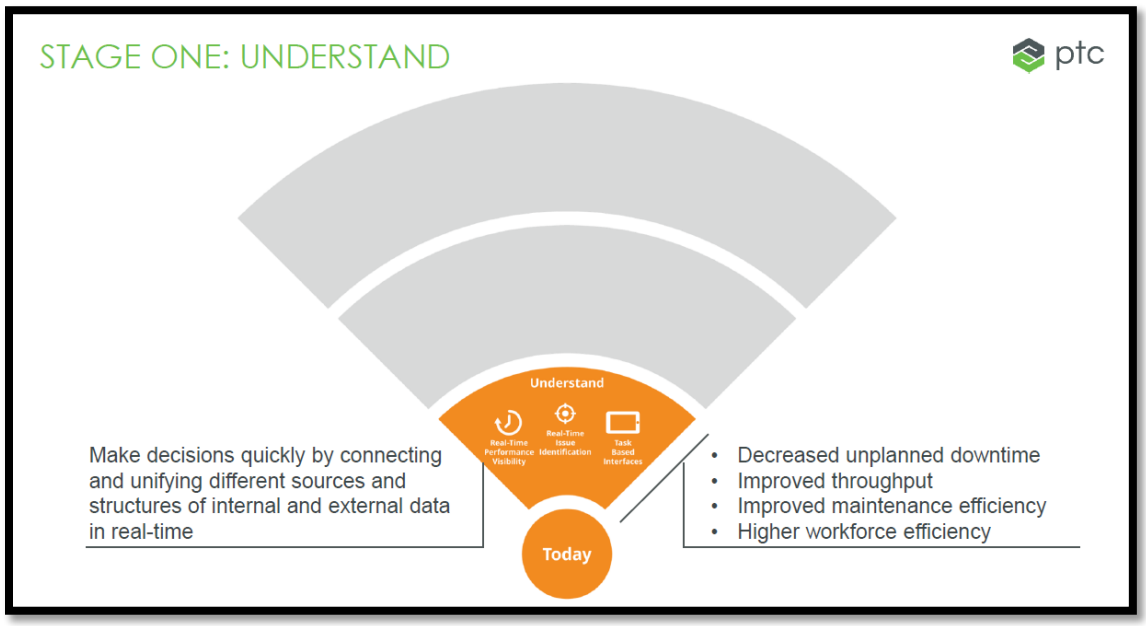


Figure 2-1: Digital Transformation-Understand Phase

The other aspect of smart, connected products that needs to be addressed is the elements of a smart setup. Smart products and systems share three main elements: physical components (mechanical/electrical parts), smart components (sensors, software, digital user interface), and connectivity components (ports, antennae, protocols, networks) (Porter and Heppelmann, 2015). Converting manual machines into smart systems will naturally require all of these components.

By providing the proper hardware, software, and communication protocols to a machining center, unified connectivity, real-time monitoring, and issue identification capabilities can be enabled. Humans would not need to constantly be at the machine pressing a button once every 10 minutes to restart a cycle. Operators would not need to spend hours stopping an entire line because of one machine unexpectedly shutting down, because the system would have told the maintenance technician days in advance that oil levels were running low, or that a belt needed to be replaced. These are merely a few examples of the power of "smart" systems with the fundamental principles of unified connectivity, real-time monitoring, and issue identification.

2.2.1 Unified Connectivity

Unified connectivity refers to a system's ability to connect and send data, whether it be from devices to other devices, or from multiple devices to a single device. The diversity of protocols and languages of disparate systems makes unified connectivity difficult for legacy equipment (Petrova-Antonova, Andreev, and Ilieva). If there is no communication being relayed between the devices involved, then there is not much else that can be done to enable other "smart" features and components. Real-time data monitoring and issue identification features associated with those machines without some form of connectivity or communication would not be realistic. However, with unified connectivity, machines can be equipped with sensors or other

digital devices to collect and exchange data. This is where the internet of things (IoT) plays a role. IoT creates a digital representation of a physical objects, machines, or processes. By collecting data from numerous different devices and/or their embedded sensors, object-to-object communication and data sharing is possible (Zhong et al., 2017). Once the connectivity gap is bridged, data can then be conveyed to and from the smart factory devices and then manipulated, allowing beneficial feedback to the supervisor (Petrova-Antonova, Andreev, and Ilieva).

2.2.2 Real-Time Monitoring

Once unified connectivity is established, the data must be put into a useable format and provide beneficial information and transparency to the end-user (Biron, Busiek, and Lang, 2018). "Real-time" refers to the actual time in which an action or operation occurs and may have its own time-precision requirements depending on the application. An example of this may be a thermometer taking temperature measurements vs. a soda-can assembly line. The real-time needs for the thermometer application would not need to be as frequent as the soda-can line. The thermometer may only need to update once every few minutes to provide useful, quality data, whereas the soda-can line updates would need to be much more frequent. Real-time monitoring is essential in being able to visualize trends and patterns and make informed decisions. Many companies with manual machines do not have such data monitoring in place simply because of the fact that no data is being generated in the first place. Having this established monitoring and visibility allow supervisors to recognize trends in production, have constant awareness of asset status, and make informed decisions (Kumar, Vaishya, and Parag, 2018). This will enable accelerated growth in the manufacturing sector and allow for more optimized processes and operations to be introduced into companies and industry as a whole.

2.2.3 Issue Identification

Issue identification takes real-time monitoring one step further. Instead of the constant need for a human to interact with the system, logic may be written that automates the system to notify an operator or a supervisor of a particular anomaly. The system has a built-in capability for responding to its own data (Roblek, Mesko, and Krapez, 2016). For example, if there is a crash on a lathe, then the voltage values being tracked are likely to spike out of a normal operating range for a brief moment. That abnormality, once detected, can then trigger an automated text or email message to the shop supervisor, prompting him/her within seconds to go out to the machine and inspect the situation.

Colfax was a company that utilized IoT to enhance their services. After establishing connectivity between their devices, they were able to allow their customers to easily identify or be alerted to deviations from normal operation, or be informed ahead of time if a system is about to fail (Thingworx, 2018). For successful smart manufacturing to be of benefit, it is essential for the system to be able to diagnose defects or other abnormalities in the production process and be able to respond accordingly (Wang et al., 2018).

2.3 Smart Implementation: Challenges

Many companies in the manufacturing industry are already moving towards smart systems and solutions to help increase efficiency, cut costs, and increase output. However, several obstacles stand in the way to building a smart setup. One of the concerns at the top of the list is operational flexibility. With the integration of new software and set ups, will the solution be scalable to the rest of the plant? This is one of the main things to consider when implementing a smart solution (PTC, 2017).

The primary focus of this project will be on how to implement a smart solution into a system that a company already has, especially if resources are limited and the machines being used are not designed to communicate in the first place. This will be simulated in an academic setting by enabling manual machines in the shop with connectivity, real-time monitoring, and issue identification capabilities, 3 of the fundamental components of "smart". In preliminary research of smart implementations, such as the 3D Systems case study mentioned earlier, machines were built already with the intent to generate and send/receive data. Case studies typically show the use of the latest cutting-edge technology to provide the "way of the future". However, many companies are still using machines that are decades old simply because they have never had problems, and there had not been a sufficiently good reason to change machining or tooling setup. There is a gap in case studies and documentation showing the conversion of manual machines (such as lathes and mills operated by human efforts) into smart systems without having to upgrade or buy new machines or tooling. Many machines simply were not designed to have connectivity capabilities. Many even have measures to prohibit such a data flow (Bates, 2018). This gap in knowledge and application in converting manual machining equipment into smart systems provides an opportunity to perform research that can be beneficial for many companies in this type of situation.

If a solution can be found, then those types of companies may be able to apply the principles learned from this research into their own circumstances, using much more affordable means (Wrenn and Thompson, 2018). Supervisors would have full-time awareness of the machines in use without even having to be on-site. The machines, equipped with these smart capabilities would be able to be more self-reliant, greatly reducing the need for constant human monitoring. Alerts of machine crashes, worn tools, low oil and coolant levels, and air leaks (to

name a few) could all be sent automatically from the machine once enabled with these smart capabilities. Manual machines are relatively unexplored when it comes to smart system conversion. This is what constitutes the focus of this study.

3 METHODOLOGY

3.1 Introduction

The following sections describe how unified connectivity, real-time monitoring, and issue identification were achieved for the lathe and the four mills. Included are pictures of physical connections, screenshots of diagrams and code snippets that were used in converting these manual machines into smart systems with these three components.

3.2 Unified Connectivity

The following sections describe in detail how unified connectivity was set up for the Kingston HJ-1100 lathe.

3.2.1 Lathe

The manual lathe that was the subject of this project was an HJ-1100 Kingston lathe, shown in Figure 3-1. This lathe was neither designed nor built with the capability to track its various electrical outputs levels and send that as data. The only type of tracking that was being used was a device that measured the *x* and *z* coordinates of the tool stock in relation to a configurable zeroed point in space. The main challenge in converting this lathe into a smart system with unified connectivity lay in measuring an electrical signal and customizing the code

needed to successfully transfer that measurement to the manufacturing apps for real-time monitoring and data analysis for issue identification.

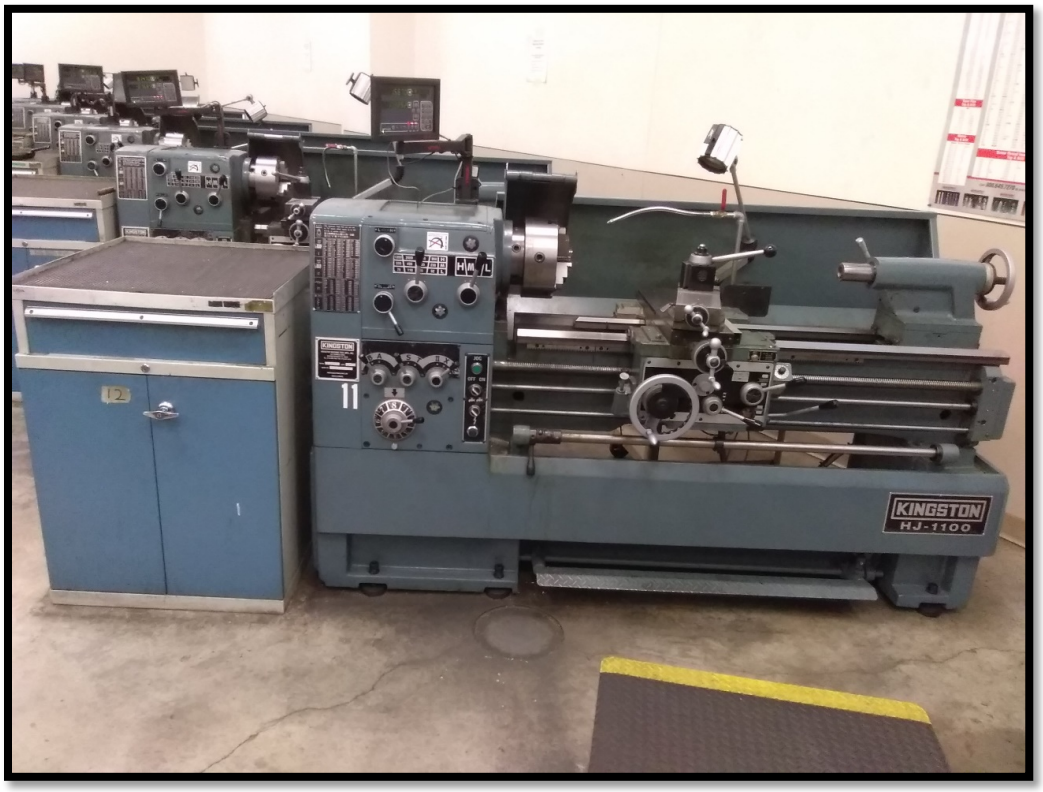


Figure 3-1: Lathe_11, Test Lathe Subject for Smart Conversion

3.2.1.1 Equipment Used

Hardware:

- HJ-1100 Kingston Lathe
- National Instruments NI cDAQ-9191 (wireless chassis)
- National Instruments NI 9225 (Isolated Analog Input Module)
- Wires to connect module to lathe electrical channels

Software:

- Laboratory Virtual Instrument Engineering Workbench (LabVIEW) 2017
- KEPServerEX (version 6.3 or later)
- Thingworx Manufacturing Apps

Links Used:

- Link for configuration and setup of the manufacturing apps:
  https://www.ptc.com/support/-/media/FF82D759E1E94A818086006219E4D3BB.pdf?sc_lang=en
- Link for a customization guide for the manufacturing apps:
  https://www.ptc.com/support/-/media/99B4D4DE487C441096DF3FC5FE36C575.pdf?sc_lang=en

3.2.1.2 Tapping into a Purely Mechanical System

Because there were no methods built into the lathe to create data or send that data to another device, it was necessary to come up with a solution to do just that. National Instruments was the company of choice because of their specialization in data acquisition, the speed at which their devices are able to acquire and send data, and because of the broad range of their presence in facilities and settings in industry across the world.

The device that was selected to aid in establishing unified connectivity on the Kingston lathe in this project was the NI cDAQ-9191. This acted as the chassis for the NI 9225 analog input module and was able to take the voltage measurements obtained by this module and send them wirelessly to a computer with LabVIEW installed. It was chosen to select a DAQ device

that could transmit data wirelessly because of the discouragement of having cords running along the factory floor. Setup of this NI cDAQ-9191 device was done using the Quick Start guide that came with the chassis. For this project, the cDAQ was set up with a unique IP address to communicate over the university Wi-Fi. Figure 3-2 shows a picture of the electrical diagram of the lathe. It depicts the numerous channels being supplied with power, and their corresponding functions.

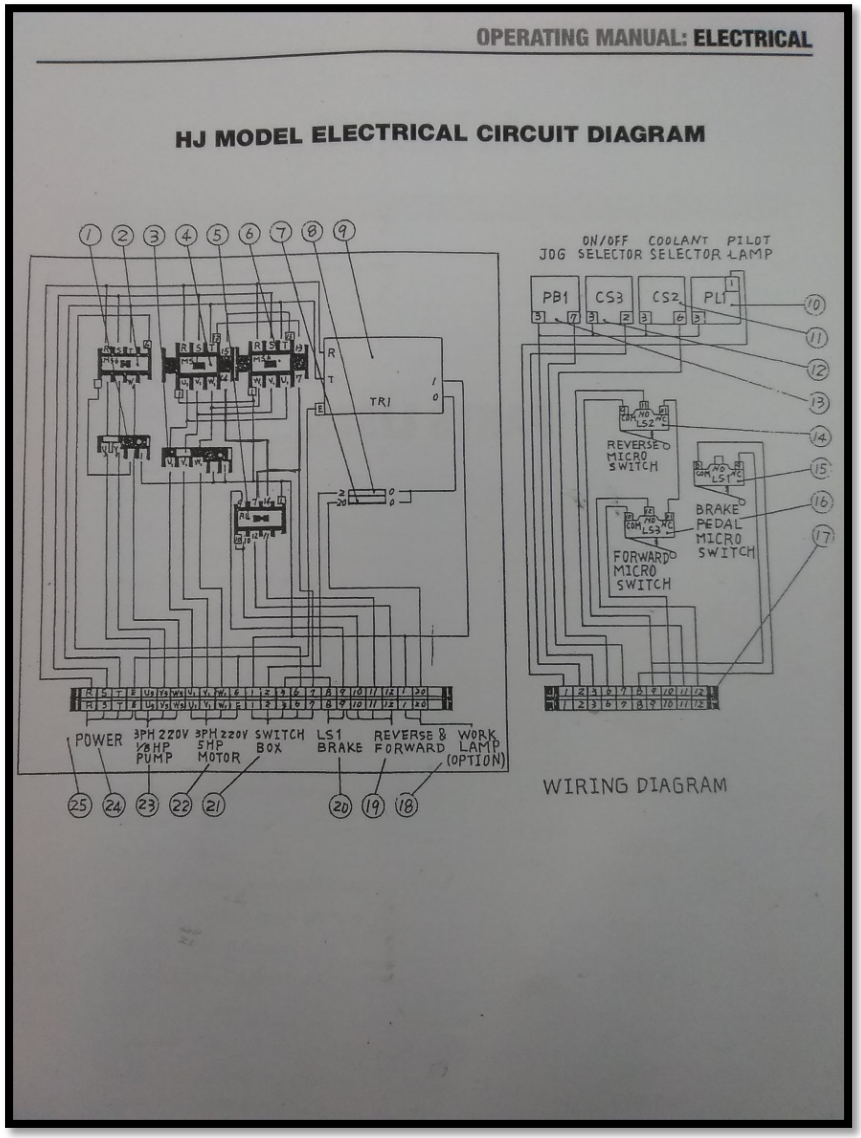


Figure 3-2: Lathe_11 Electric Diagram

The cDAQ-9191 has three ports from which it can send data. Two of these ports were used in this project. The pieces of information that were selected to be tracked were the channels associated with the on/off selector (going through Port 0 on the cDAQ) and the channels associated with the motor (going through Port 2 on the cDAQ). These physical connections are shown in Figure 3-3. From these ports, the voltages would change according to the machine's status and would then form a predictable pattern off of which the state of the machine could be determined. These voltage values would be sent to a computer with LabVIEW (Laboratory Virtual Instrument Engineering Workbench) installed.

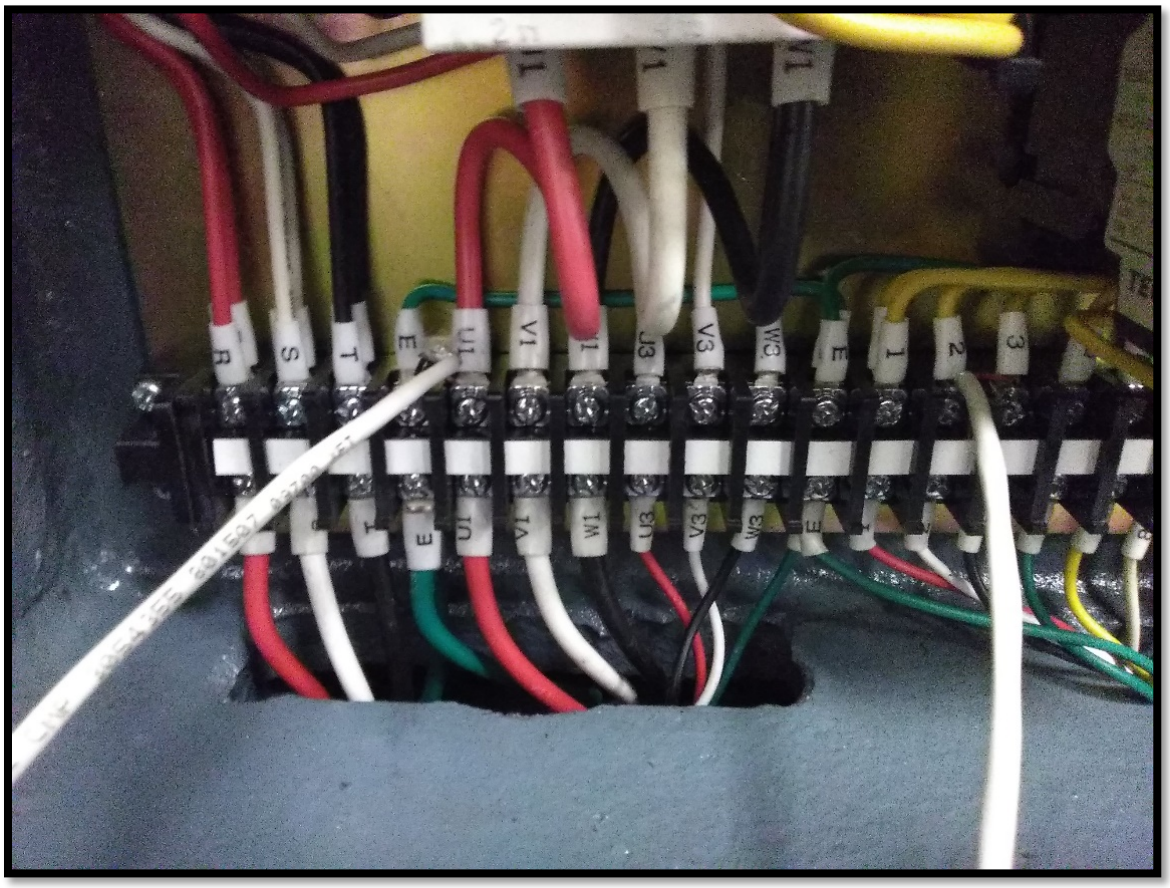


Figure 3-3: Wires Connecting to the ON/OFF Selector and the Motor Channels

3.2.1.3 LabVIEW to KEPServerEX

LabVIEW is a system-design platform that uses visual programming to monitor and control data being acquired by acquisition devices. In this project, a National Instruments device, was used in conjunction with LabVIEW. By configuring the NI cDAQ-9191 to communicate with a computer with LabVIEW installed, that data could now be acquired, altered, and forwarded on to KEPServerEX. From KEPServerEX, the data could then be sent to the manufacturing apps. The biggest challenge, however, was getting the acquired data from LabVIEW to KEPServerEX so that all of the machines in the smart system setup could be streamlined from one source into the manufacturing apps.

In the LabVIEW program that was created and used for this project (shown in Appendix A: LabVIEW Code) several things were done to the data being acquired before it was sent to KEPServerEX. The positive peaks for the voltage measurements were taken, giving a true value of what the voltage actually was instead of a near-zero, averaged measurement from equally positive and negative voltages. The rate at which the samples were taken was 1000 Hz with 100 samples to read. The values on these two voltage ports were then expressed as doubles, averaged, and then converted into strings for ease of use in other programs. Simultaneously, the doubles being used were evaluated to determine if the machine spindle was running or not. If the machine spindle was running, then a timer turned on and updated the accumulated run time. This spindle running time was then sent with the other two voltage values, as a string, to the rest of the program.

In sending the data to KEPServerEX, it was necessary to write a customized LabVIEW program that had the ability to send its acquired data to KEPServerEX. This was done using an Open Platform Communication Unified Architecture (OPC UA) connection, with KEPServerEX

acting as the client, and LabVIEW acting as the server. OPC UA is a machine to machine protocol developed by the OPC Foundation and is used for industrial automation. Because of its flexibility in being able to work with many different types of machines and not being tied to any one machine or language, it was found to be an appropriate fit for smart conversion of the Kingston lathe. Fortunately, National Instruments has already recognized the trend towards OPC UA connections and in fact has example VIs that establish this kind of connection. This example VI was used as the basis for connection and was then customized for this particular project application. An in-depth depiction and description of the LabVIEW program used in this smart setup is given in Appendix A: LabVIEW Code.

Before the data could be brought into KEPServerEX, however, the proper channel, device, and tags had to be created with the proper parameters. Details of the channel, device, and tag configurations inside KEPServerEX will be given below.

Pre-Work for Setting up an OPC UA/KEPServerEX Connection-

- Using/deploying OPC Servers, DSC, and LabVIEW OPC UA Toolkit: http://www.ni.com/product-documentation/54990/en/
- Download KEPServerEX (6.3 is the version that was used for this project)

The demo-version of KEPServerEX only runs for 2 hours at a time. If possible, obtain a professional license for an instance of KEPServerEX.

For this connection to occur, make sure that the OPC UA Toolkit is installed with LabVIEW 2017, and that the OPC UA Client driver is installed with KEPServerEX.

- Open the newly installed KEPServerEX program
- In the left hand panel, right click on “Connectivity” and click “New Channel”

Channel Creation Wizard Settings-

A representation of all of the settings used to configure the channel inside KEPServerEX is given in Figure 3-4. Many of these settings are specifically based off of the research computer from which this LabVIEW instance was hosted. When replicating this experiment, these settings may be used, but it may be necessary to open a support case with National Instruments and Kepware in obtaining the correct endpoint URL and driver specifications.

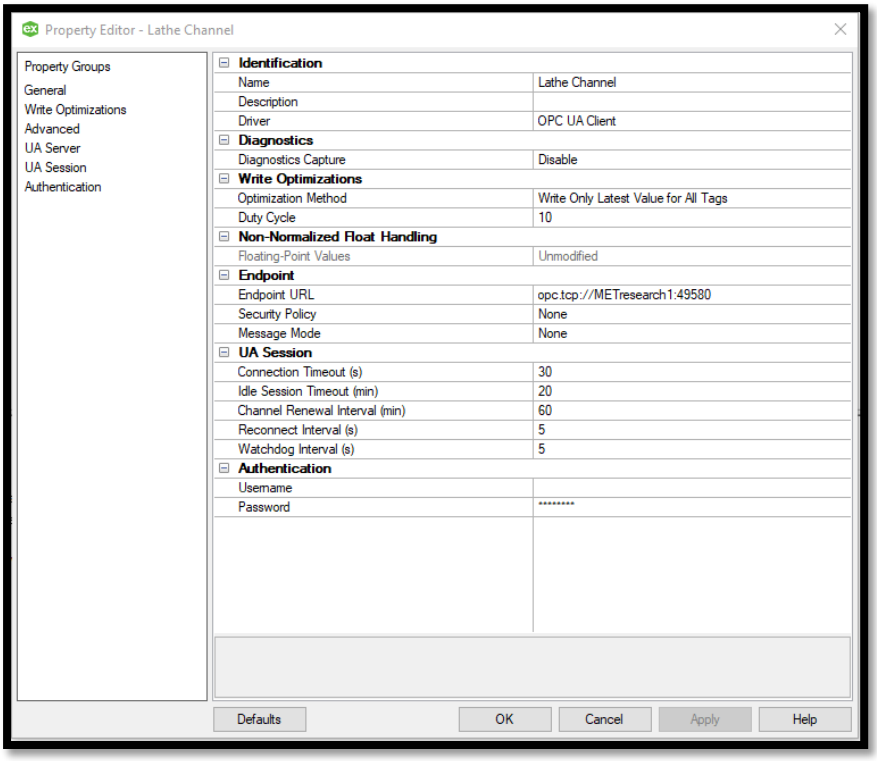


Figure 3-4: Summary Page of New Channel Settings

Device and Tag Creation Wizard Settings-

Upon creation of the channel, a device was set up with the needed parameters and settings. Figure 3-5 shows a list of the parameters used in the creation of the device in this project named Lathe11.

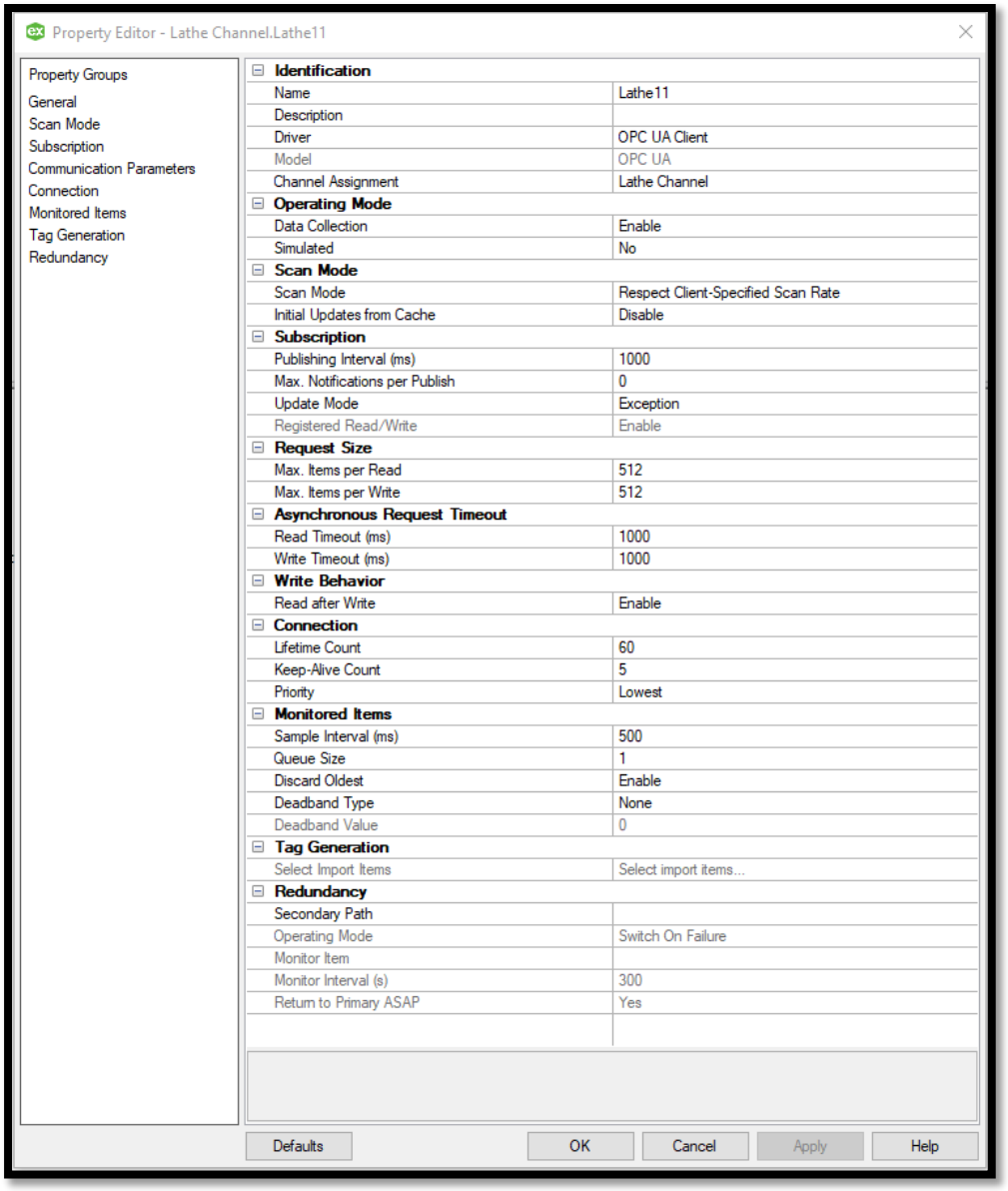


Figure 3-5: Summary of Device Creation Settings

After the device was set up, tags were created to reflect the incoming data from the machines or communication cards. Right-click on the newly-created device and select New Tag. Figure 3-6 is a representation of what information is needed for tag creation. Repeat this for as many tags as needed. The specific address will be needed in order to successfully establish a good connection with live data updating in real-time.

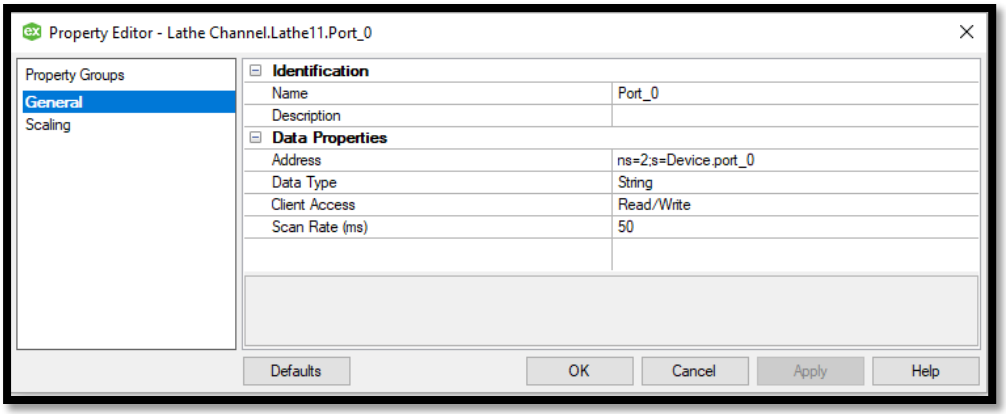


Figure 3-6: Summary of Tag Settings

In order to find the address for this specific tag within the LabVIEW program, probe the pink wire going into the NI OPC UA Server.lvlib:Write.vi from the corresponding NI OPC UA Server.lvlib:Add Item.vi file while the program is running.

Run the Quick Client with the LabVIEW program running in the background to check and make sure that the data is coming in from LabVIEW and is updating in KEPServerEX.

3.2.1.4 KEPServerEX to ThingWorx Manufacturing Apps

Once the data is seen updating in the Quick Client, it is ready to be sent to the ThingWorx Manufacturing apps. From this point, use the Configuration and Setup Guide for the Thingworx Manufacturing Apps (version 8.3) to create a digital twin of the lathe and bring in the real-time data of the smart lathe setup. Once the data comes into KEPServerEX, the connection between the lathe and the manufacturing apps is essentially seamless and reliable unified connectivity is established. KEPServerEX was built to have that connectivity with ThingWorx and allows robust solutions to be integrated between completely different devices, as will be shown.

3.2.2 Mills

3.2.2.1 Equipment Used

Hardware:

- 2 ICC ETH-200 Communication Cards (Ethernet Multiprotocol Network Gateway)
- Tosvert VF-S7 (Industrial inverter used on Mill 1)
- Tosvert VF-S9 (Industrial inverter used on Mills 2-4)
- 6 Cat 5 Ethernet cords
- 2 Power Supply 9VDC Barrel Jack cords

Software:

- KEPServerEX (version 6.3 or later)
- ThingWorx manufacturing apps
- Embedded webserver (configured using the ETH-200 cards)

3.2.2.2 VF-S7 to ETH-200 Communication Card

The protocol that the VF-S7 inverter uses is an older version of Toshiba protocol. When looking into the drivers that KEPServerEX supports, this Toshiba protocol was not supported by current drivers. A change in the protocol had to be made. After some research and communication with Toshiba representatives, the recommended communication card that would help to convert the older Toshiba protocol into a more usable and compatible protocol (such as Modbus TCP/IP) was the ICC ETH-200 Communication Card. Standard Cat 5 Ethernet cords

were used in the setup of the mills to connect the inverter to the communication card, and the communication card to a computer with KEPServerEX. The final setup of the mill connection is shown in Figure 3-7. It shows the inverter being connected to the communication card via serial connection. The card is then connected on the bottom to the computer with KEPServerEX installed.

Figure 3-7: Communication Connection Between the Mill and KEPServerEX

The communication card may be powered in one of two ways: by an auxiliary power cord, or by the inverter on the mill itself. For this project, it was decided to go with the auxiliary power cord because the machines may be in an e-stop condition, which cuts off power supply to

the machine and subsequently, the card. One cord comes from each machine's inverter (from the Ethernet jack) and plugs into the ASD1 serial port. Once the milling machine is connected to the card, and is out of an e-stop state, a green LED light on the ASD1 port will light up, signaling that the machine is connected. Another cord then goes from the Ethernet/IP jack on the side of the communication card. An orange light will begin to flash orange, signaling that this connection has been properly established.

3.2.2.3 Embedded Web Server Configuration

From the Ethernet connection, the ports, protocols, and registers may be configured via an embedded web server. Here is a link to the ETH-200 user manual that will help with the setup:

- http://www.iccdesigns.com/products/gateway/eth200/documents/ETH-200%20V1.130%20User%27s%20Manual.pdf

  Below are the specific instructions on the steps to integrate the ETH-200 into this project.

- Type the default factory IP address (pg. 28 of the user manual pdf) of the ETH-200 into a web browser. In this application, it was 192.168.1.100.

- Authentication will be required. For the first time setting up the card, the following credentials are used:

  - Username: admin

  - Password: (nothing, simply press enter)

- Once inside the web server, the IP address may be reconfigured to be unique.
- Set the correct date and time
- You may set up a new admin and user username and password if desired
- Enable the necessary connections for your application
    - For this project, the drive going to the ETH-200 is through the ASD1 and ASD2 ports via a CAT 5 Ethernet cable connection, and another Ethernet cable going from the Ethernet/IP port to the Ethernet port on the laptop. I enabled the ASD1 port and Ethernet/IP port and disabled all others on the web server page

After each change to the web server interface, if there's a "submit" button in the section you are changing, click submit to update the parameters. Figure 3-8 provides a screenshot of the interface and the configurations of the ports used for this project:

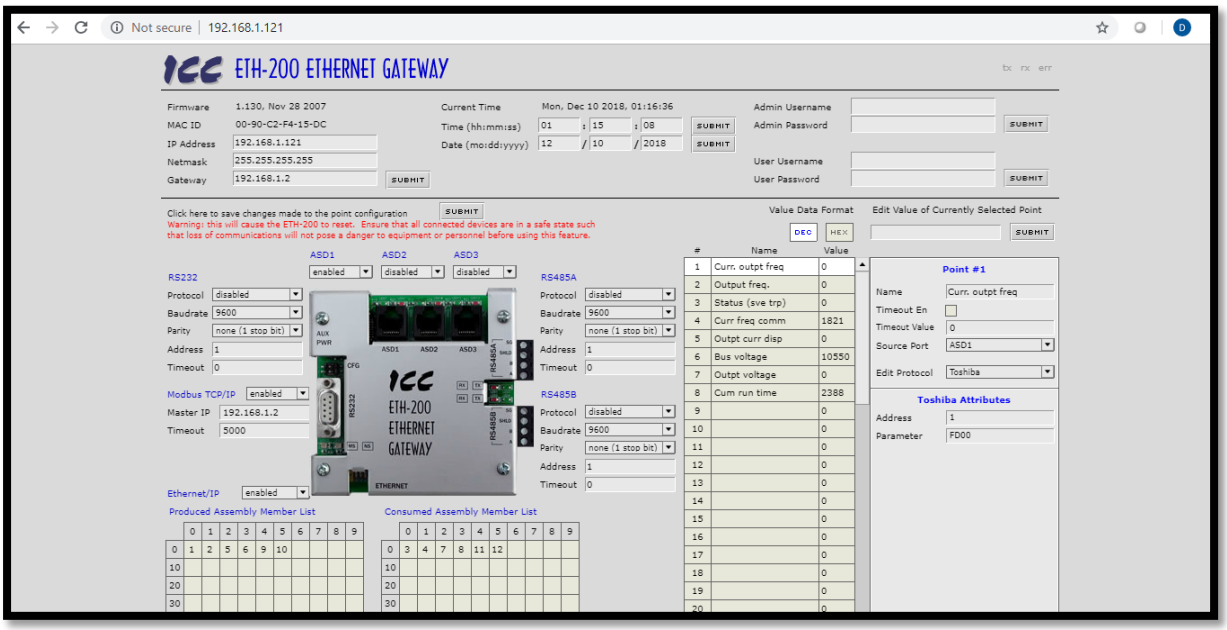


Figure 3-8: Embedded Web Server on the ETH-200 Communication Card

3.2.2.4 Point-To-Point Configuration

Now that the basic communication channels are set up appropriately, configuration of the ETH-200 to monitor the right points/tags/addresses within in the VF-S7 drive is the next step in the process in achieving unified connectivity: the card must look for the specific pieces of information that are desired. This is done by setting up points with the correct settings, channels, and protocols and is referred to as point-to-point configuration, as shown in Figure 3-9.

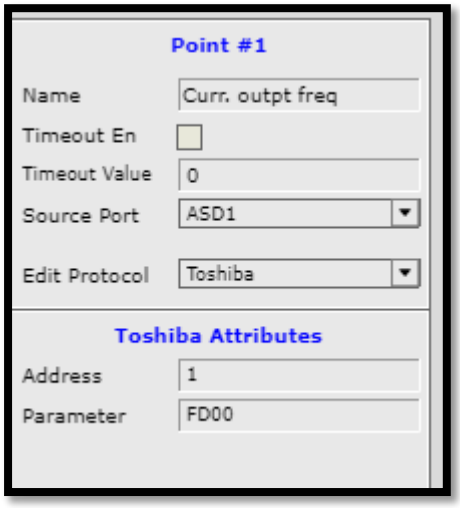


Figure 3-9: Screenshot of a Point-to-Point Configuration for a Point on Mill 1

- Address: which device (all of these will be a "1")
- Parameter: these are the tags/registers listed in the VF-S7 manual. Because it is desired to only monitor settings, and not change them (for liability reasons) only "read-only" tags will be used. These are found in the table(s) on pg. 35-36 of the Toshiba VF-S7 Serial communications option manual. The VF-S9 manual is essentially the same as the VF-S7 manual, but a link is provided below in the event of future explorations of other data and aspects of the machine:

- VF-S7 manual: http://www.inverter-plc.com/toshiba/VF-S7_Serial_Communication_Function_Manual_e6580720.pdf
- VF-S9 manual: http://www.efesotomasyon.com/toshiba/VF-S9_Communications_Function_e6581139.pdf

- Repeat this process for each piece of desired information
- Click submit to refresh the changes made

There are up to 100 points that may be configured and collected from connected devices. If multiple machines are being connected to the same ETH-200 card, the second, third, etc. machine points may be listed further down in the column. Just ensure to specify the port from which the information will be pulled in the point-to-point configuration section and refresh upon completion.

3.2.2.5 ETH-200 Communication Card to KEPServerEX

- Download KEPServerEX (6.3 is the version that was used for this project)

Ensure that the installation includes the Modbus suite drivers.

The demo-version of KEPServerEX only runs for 2 hours at a time. If possible, obtain a professional license for an instance of KEPServerEX. The professional version of the license provides constant communication with all of its devices without the need to restart the program every two hours.

- Open the newly installed KEPServerEX program
- In the left hand panel, right click on “Connectivity” and click “New Channel”

Channel Creation Wizard Settings-

Similar to the lathe, channels, devices, and tags had to be created inside KEPServerEX. In this project, one channel with the Modbus TCP/IP Ethernet driver was configured. Figure 3-10 shows the channel configuration for connecting to the four mills used in this project. Once the settings are configured as listed above, click Finish.

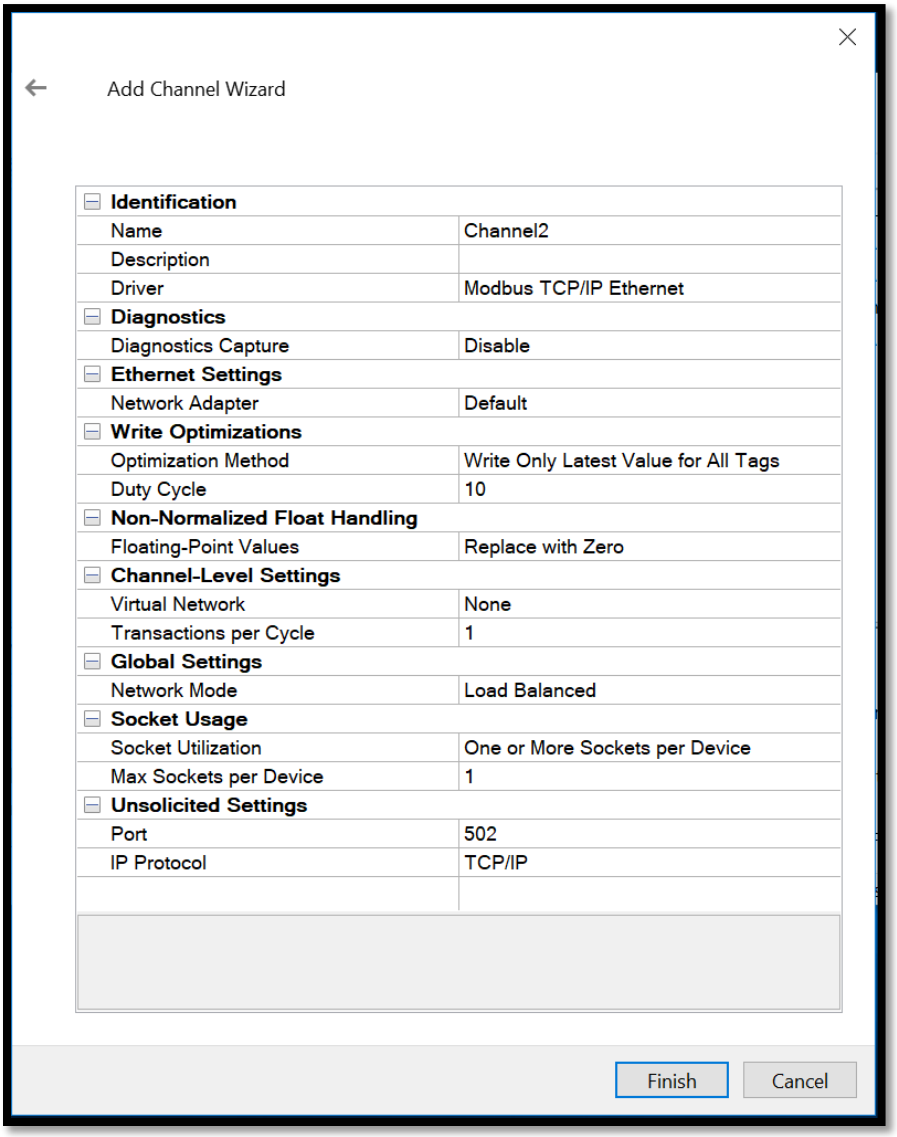


Figure 3-10: Summary Page of New Channel Settings

Device Creation Wizard Settings-

After creating the channel, the “eth200 device” was created (click on “click to add new device” in the left panel). Here are the different parameters that were set for this device. Not all of the settings from each tab were able to be shown, so screenshots for each individual tab/step through the wizard are provided for eth200 device creation:

Figure 3- 11 shows the settings used for the General tab upon device creation. Here the driver being used is represented along with the ID with the corresponding IP address of the machine/communication card. Additionally, one may enable the data to be collected or not, as well as having the device or data collection be simulated or not. The type of driver being used for this device follows the channel driver (Modbus) and specifically uses the Ethernet form of Modbus TCP/IP.

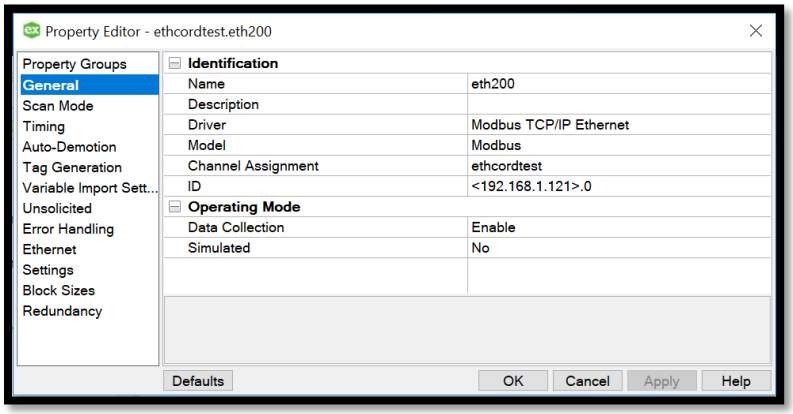


Figure 3- 11: General Tab Settings

Figure 3-12 shows the scan rate that will be used for the device that was set up for the ETH-200 communication card. It is left at the default rate, updating at the rate of which the client (the mill driver) is pushing out information. The mill driver is set to send the data that is being

requested at about 40 milliseconds per push. The rate of scan inside of KEPServerEX may be adjusted based off of what is desired, but is limited to a scan rate of 10 milliseconds.

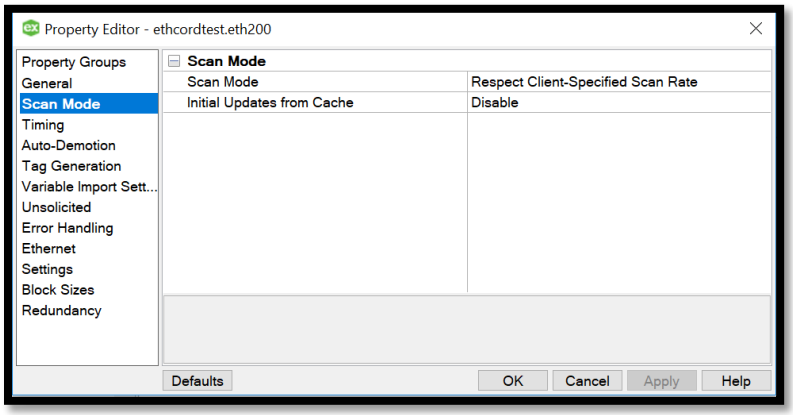


Figure 3-12: Scan Mode Tab Settings

Figure 3-13 depicts the timing settings that were used on the device created in KEPServerEX. These were the default settings that were provided in the configuration pop-ups.

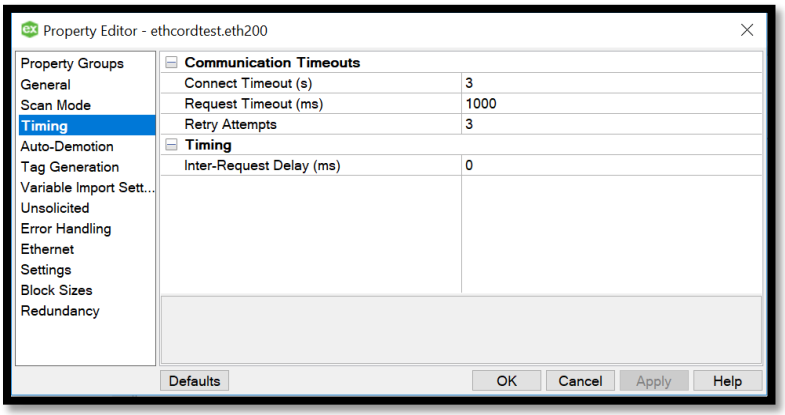


Figure 3-13: Timing Tab Settings

Figure 3-14 shows the setting that is used for this device configuration. Auto-demotion refers to what KEPServerEX will do when the device is unresponsive. If enabled, KEPServerEX

will place the device off-scan for a set time in order to optimize its communication with other devices. This device will continually be scanned, even when unresponsive, in order to ensure the most consistent communication possible.

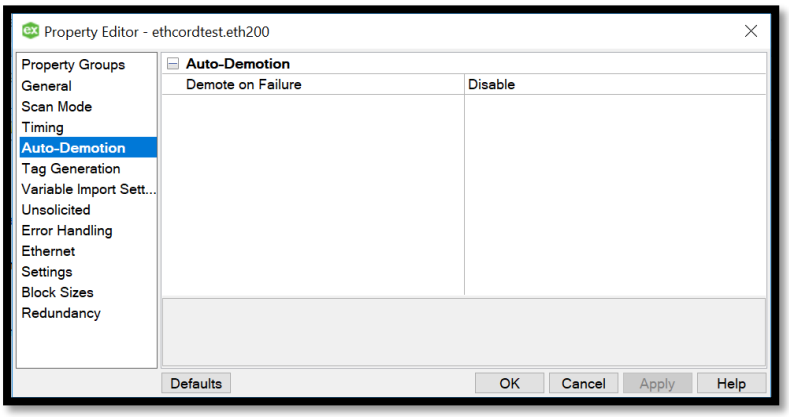


Figure 3-14: Auto-Demotion Tab Settings

Figure 3-15 defines how tags can and will be generated. For this project, is was chosen to manually create the tags in order to be selective in the information being tracked and received.

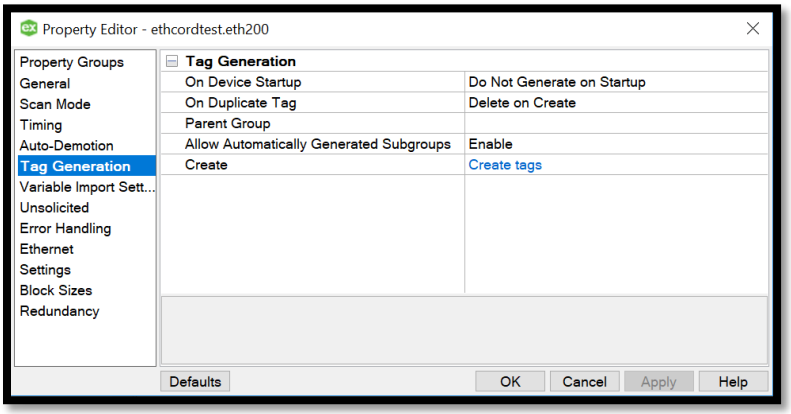


Figure 3-15: Tag Generation Tab Settings

Figure 3-16 shows the settings used on the variable import settings tab. The variable import settings tab specifies the parameters and types of files acceptable for importing tags.

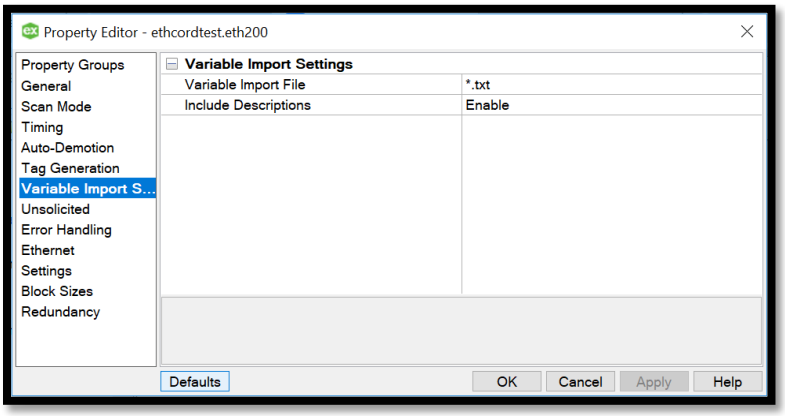


Figure 3-16: Variable Import Settings Tab Settings

Figure 3-17 shows the settings used on the unsolicited tab of device creation. If disabled, all tags have an initial value of 0 and an OPC quality of Good. If enabled, all tags have an initial value of 0 and an OPC quality of Bad.

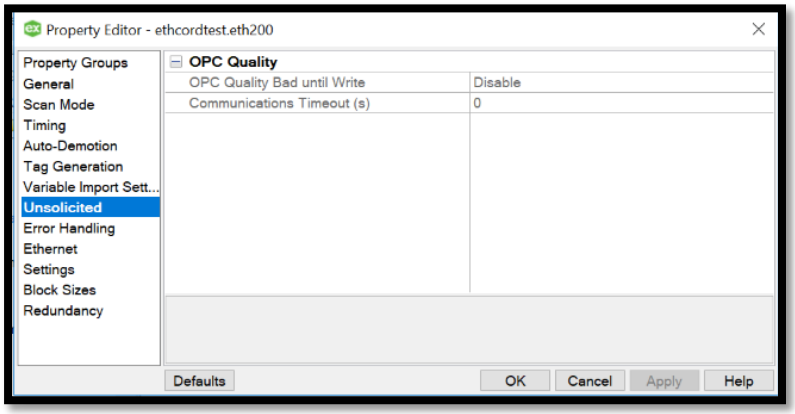


Figure 3-17: Unsolicited Tab Settings

Figure 3-18 shows the settings used for the error handling tab. When enabled, the driver will stop polling for data if there is an illegal address or illegal data involved. If disabled, the driver will poll despite such errors.

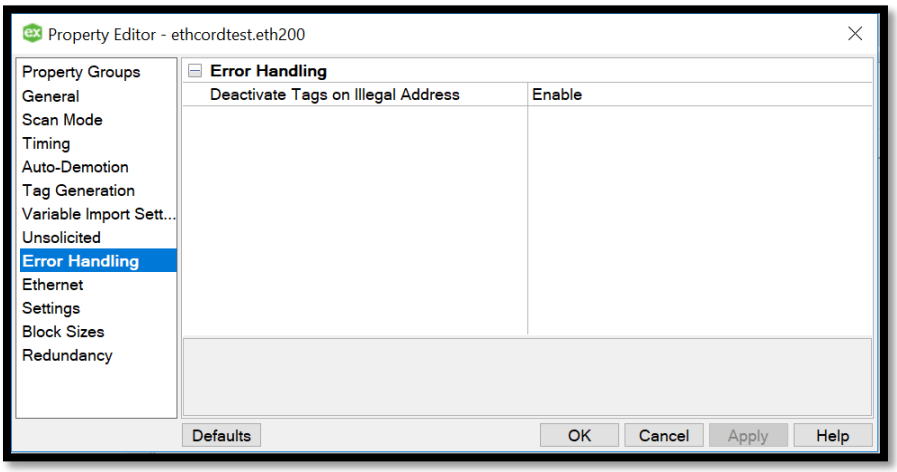


Figure 3-18: Error Handling Tab Settings

Figure 3-19 shows the settings used for the Ethernet tab in device creation, with the associated port and IP protocol.

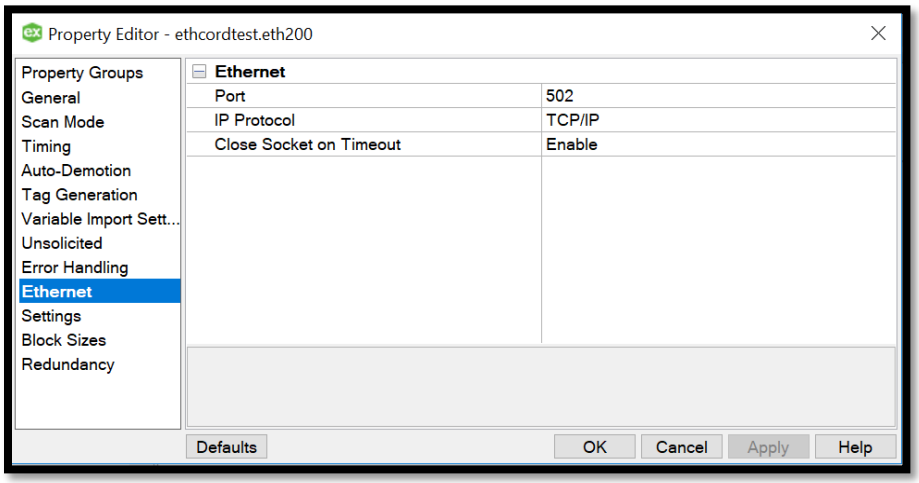


Figure 3-19: Ethernet Tab Settings

Figure 3-20 shows the enabled/disabled values used for this project in the settings tab inside device creation. The defaults were used for this pop-up.

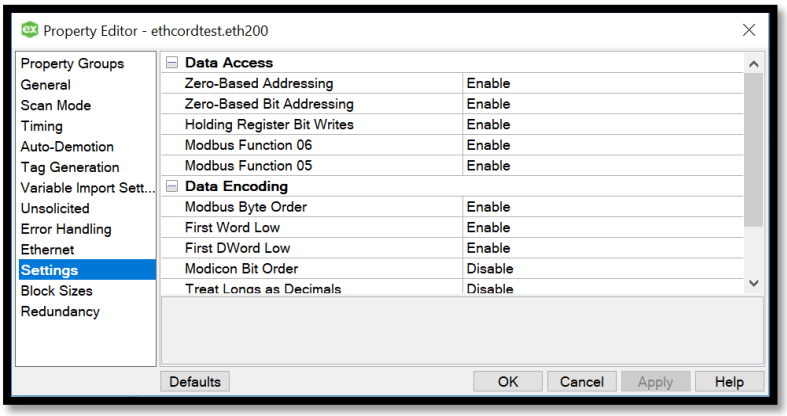


Figure 3-20: Settings Tab Settings

Figure 3-21 shows the settings used in the block sizes tab of device creation. The default settings were used in this configuration.

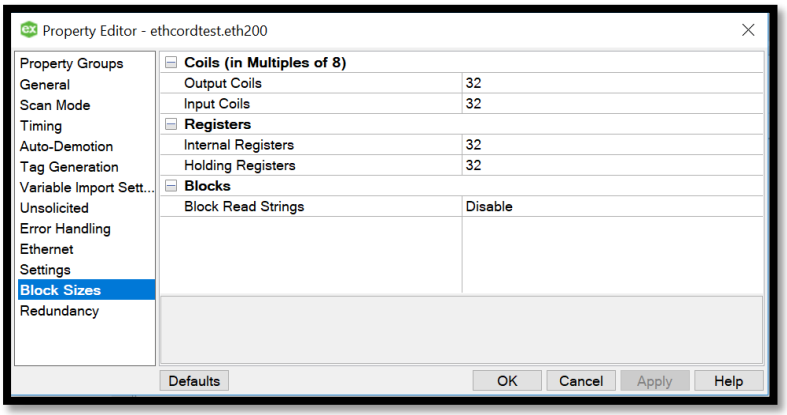


Figure 3-21: Block Sizes Tab Settings

Figure 3-22 shows the settings used in the redundancy tab of device creation. However, this is only available with an additional Media-Level plug-in. Consequently, the options are

grayed-out, and the defaults were used for this application, but there were no negative side-effects as a result.

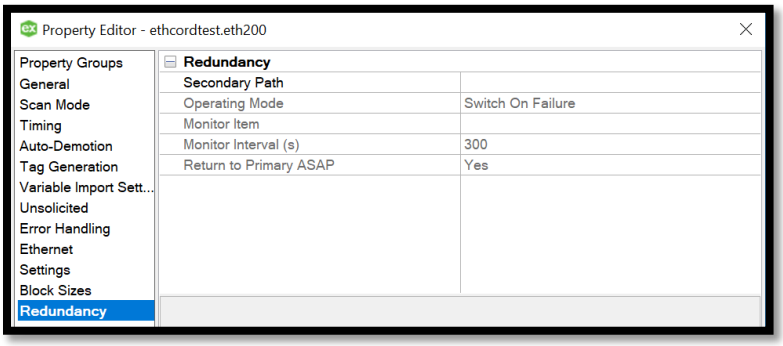


Figure 3-22: Redundancy Tab Settings

Tag Creation-

- Right click on the device that was just created. Click New Tag. Figure 3-23 is a screenshot of the inputs for a new tag for the eth200 device. No changes were made to the default values in the Scaling tab, so no picture is provided for the scaling tab. It is recommended that the tag name match the name given in the embedded web server for consistency purposes. Click OK

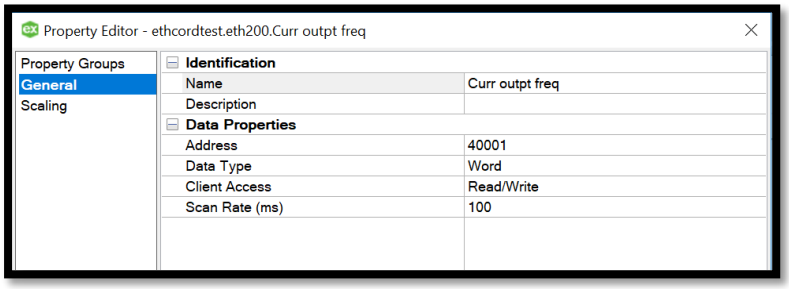


Figure 3-23: Mill Tag Creation

Modbus TCP/IP addresses start in the 40000 range. For this application, to get the correct information from the ETH-200 communication card, go to the embedded web server, look at the row number for the parameter needed and add 40000 to it (shown above). Ex: 1 (on the web server) converts to an address of 40001 in KEPServerEX.

If establishing a KEPServerEX channel, device, and tags for the first time, it is recommended that you call a PTC/KEPServerEX representative to help navigate the numerous parameters and settings needed for a proper connection.

- Run the Quick Client to make sure that data is updating properly and in real-time. Figure 3-24 shows a representation of what the channel, device, and created tags look like as well as where to go to run the Quick Client and navigate to the newly bound data

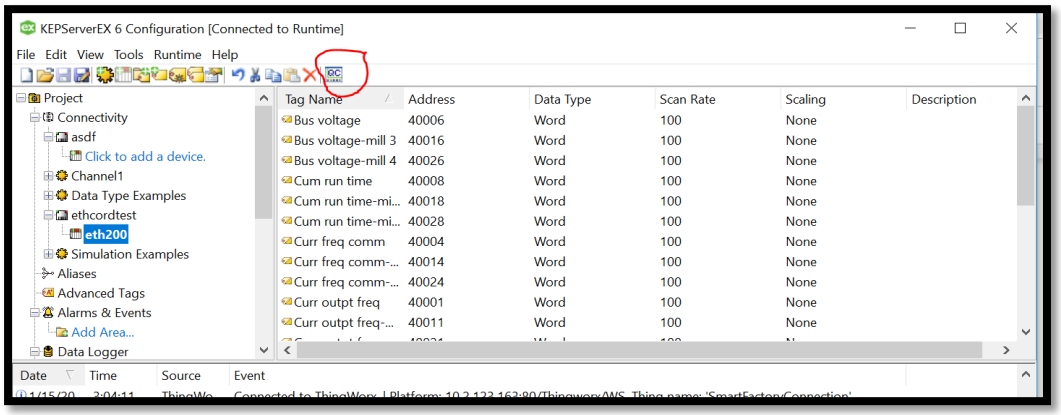


Figure 3-24: Channel, Device, and Tag Creation, Quick Client

When the popup window comes up, scroll down to the device with the newly created tags (ethcordtest.eth200 in this application). On the right, the tags will be shown updating in real-time.

If you get a “Bad” connection, check your physical connections and your channel, device, and tag settings to make sure that everything is configured correctly, then try running Quick Client again.

3.2.2.6 KEPServerEX to ThingWorx Manufacturing Apps

- Install the ThingWorx Manufacturing Apps as an extension to ThingWorx Composer. Make sure that the versions of ThingWorx Composer and the apps are compatible with each other. This project used ThingWorx Composer and apps version 8.3
- Walk through the steps of the Configuration and Setup Guide to set up equipment with additional properties tied to the tags with the real-time data being tracked from the ETH-200 communication card

3.3 Real-Time Monitoring

3.3.1 Lathe

Now that raw data is coming into KEPServerEX and the ThingWorx Manufacturing Apps, it becomes expedient to convert that data into something useful for a production supervisor or manufacturing engineer. Two numeric voltage values will not mean much to anyone. This data will need to be interpreted by the smart system in which it plays a part. After conversing with the shop supervisor about the lathe, several pieces of information to monitor in real-time were determined: machine status (machine on/off, spindle state, and brake state, forward/reverse rotation of spindle), tracking the spindle-on time, and checking to see if there was a crash on the machine.

3.3.1.1 Spindle Time

The spindle time was another desired piece of information to be monitored in real-time and displayed. This was made easiest by writing the logic in the LabVIEW program being run in the background of the smart setup. The LabVIEW code needed to obtain and send this piece of data to KEPServerEX over a OPC UA connection is given in Appendix A: LabVIEW Code.

Once the spindle time was brought into KEPServerEX and subsequently the manufacturing apps, it can be added as an Additional Property within the Configuration and Setup tile under the Lathe_11 asset. All properties brought into the manufacturing apps are then monitored in real-time, updating every few seconds.

3.3.1.2 Machine Status

Once all of the desired properties are in either the apps or KEPServerEX, the machine status may be defined based off of the brought in properties (Figure 3-25). For instructions on how to properly configure the status of the asset or machine, refer to the Configuration and Setup Guide for the ThingWorx Manufacturing Apps. The statuses defined in the manufacturing apps were set up based off of the dynamic values of the tags being pulled in from KEPServerEX. As these tags change, so will the statuses. For example, the machine will be considered to be in a "running" state for the lathe if both the MachineState and MotorState tags are evaluated to be "true". For the Planned Downtime status, this can be defined in the General Information tab inside of the manufacturing apps and can be set for whichever time the operator supervisor decides.

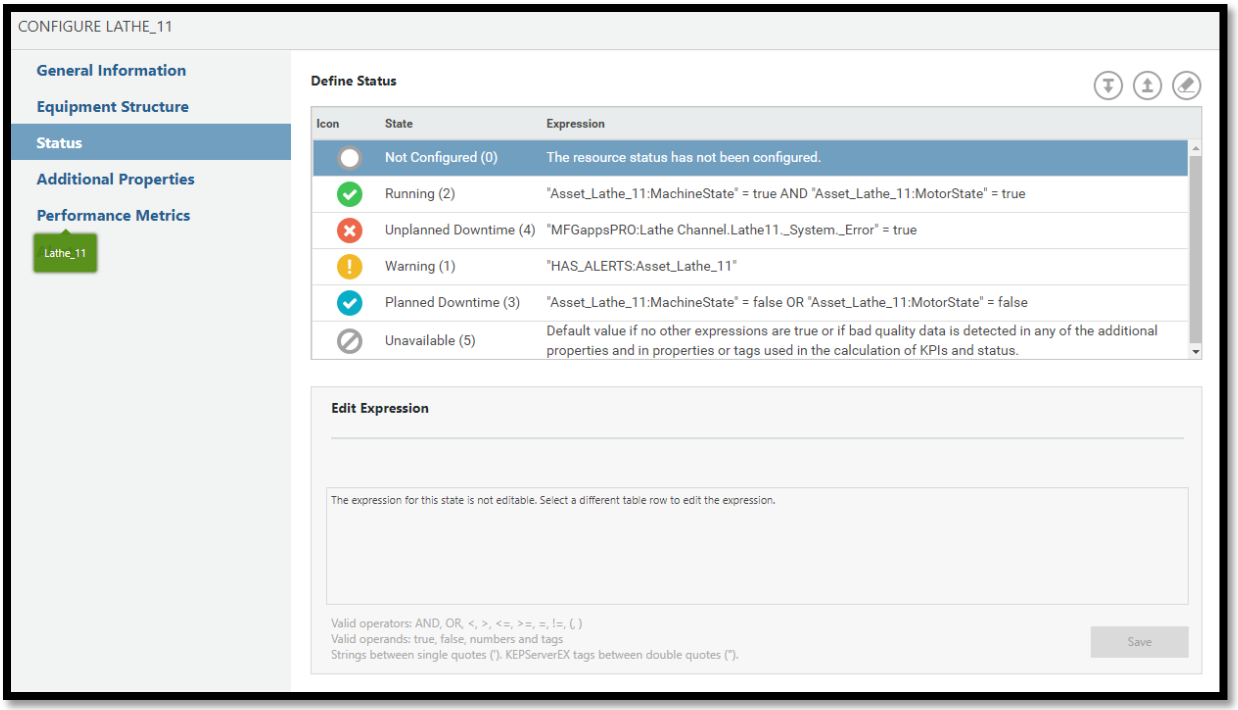


Figure 3-25: Status Configuration for Lathe_11

3.3.2 Mills

3.3.2.1 Machine Status

Similar to the lathe status configuration within the Configuration and Setup tile of the manufacturing apps, the mills can be individually defined for each different type of machine state (Figure 3-26). However, the mills were able to keep a more comprehensive track of the machine status because of the information already on the drive. Unlike the lathe, the mills started the communication process with a protocol. The protocol had to be converted to a more useable format, but there was something preexisting in place.

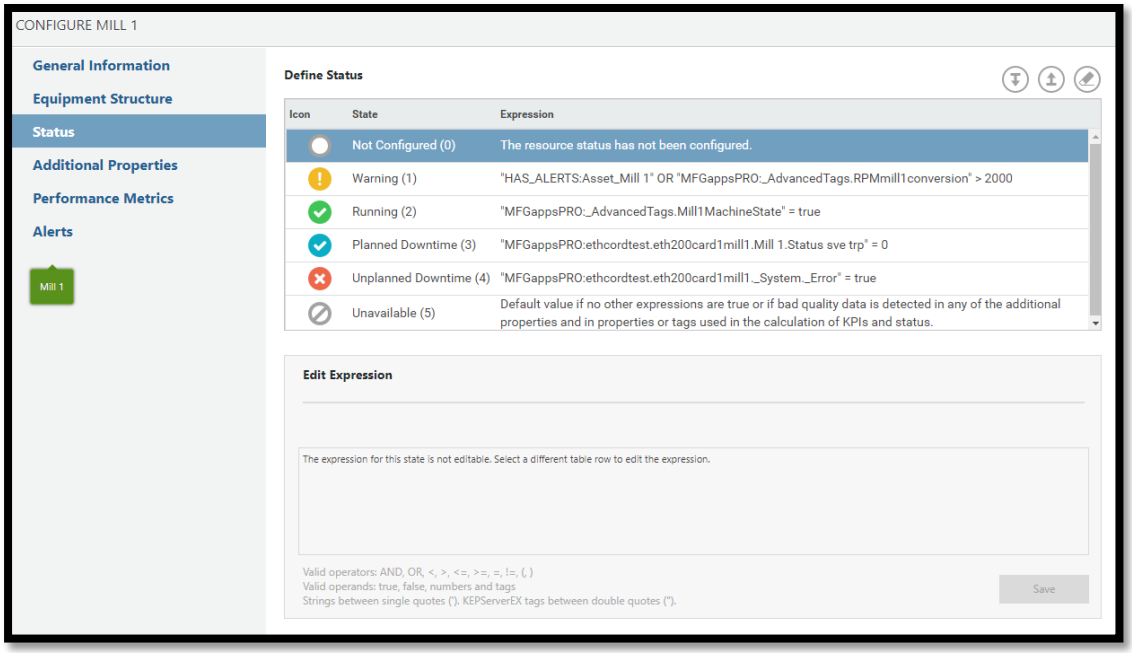


Figure 3-26: Status Configuration for Mill 1

3.3.2.2 RPM

The RPM of the mills did not come as a direct result of the information provided from the inverter. Rather, it had to be derived from said information, specifically the Current Output Frequency tag. This was accomplished inside of the KEPServerEX opf file by creating a derived tag based off of the Mill 1 current output frequency. This is done by right-clicking the Advanced Tags option on the left hand-side of the KEPServerEX window and clicking on Derived Tag. Figure 3-27 shows a depiction of one of the derived tags used for Mill 1. Other derived tags were also configured during the setup process. This includes the derived tags needed for the other mills in the smart factory as well as the machine status derived tags that would help to discern the machine status for the lathe.

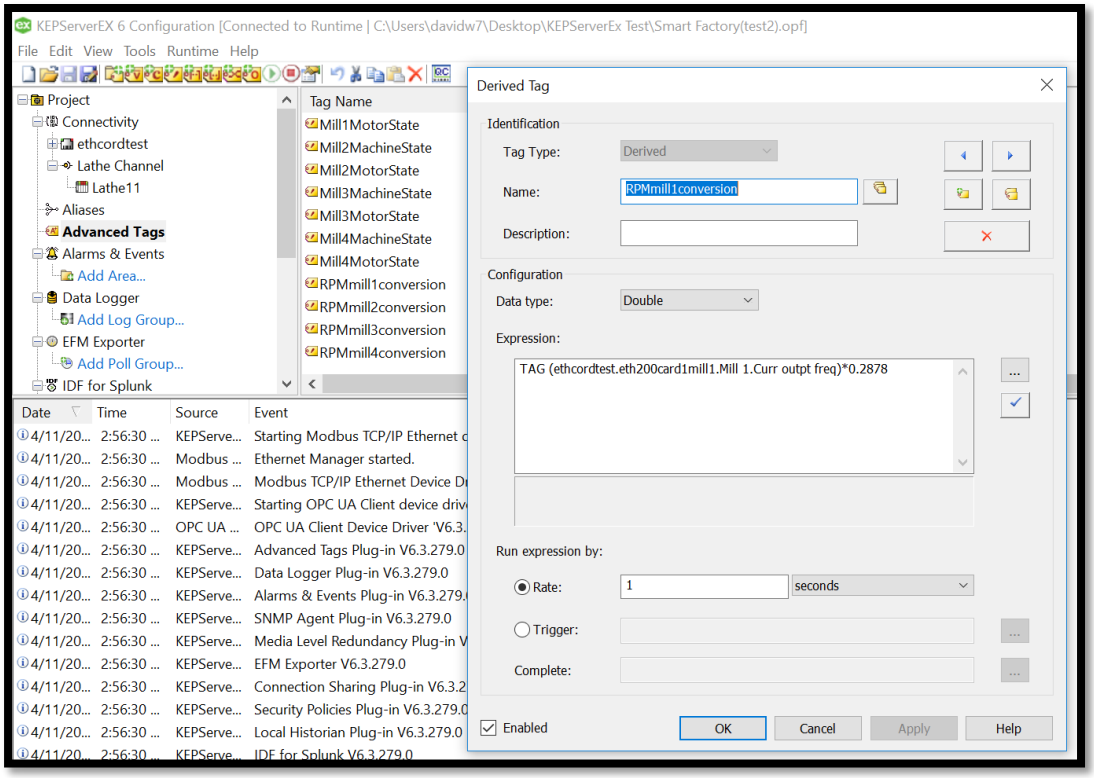


Figure 3-27: Real-Time Monitoring of RPM from a Derived Tag

3.4 Issue Identification

3.4.1 Setting Up Users with Email and Text Notification Capabilities

This step is done using the Configuration and Setup Guide for the Thingworx Manufacturing Apps 8.3. The link to this guide is given above in the section regarding KEPServerEX to ThingWorx Manufacturing apps. The process of doing this takes place inside of the Configuration and Setup Tile inside of the manufacturing apps. Once there, go to the Users tab. Here, one is able to set up a User or edit an existing User and allow them the appropriate levels of access and authorization.

3.4.2 Lathe

3.4.2.1 After-Hour Operation

After the users are configured in the manufacturing apps, services can then be written in ThingWorx to scan and see if any of the machines are still operating after specified hours. There were three Things in Composer that were created to provide this after hour alert capability: AfterHoursScan, EnableAfterHoursScan, and DisableAfterHoursScan. All of these Things inherited the Scheduler base thing template in the General Information tab upon creation. Once saved, these schedulers may then be scheduled for a particular time in the Configuration tab. The schedule property must be provided in CRON format.

After the schedule is configured, a subscription is fired based off of the scheduled event that calls and activates a service to perform a function. In this project, the name of that service is called ScanMachines and simply sends an email and a text to the specified supervisor in the event that one of the machines is operating after normal hours (Figure 3-28).

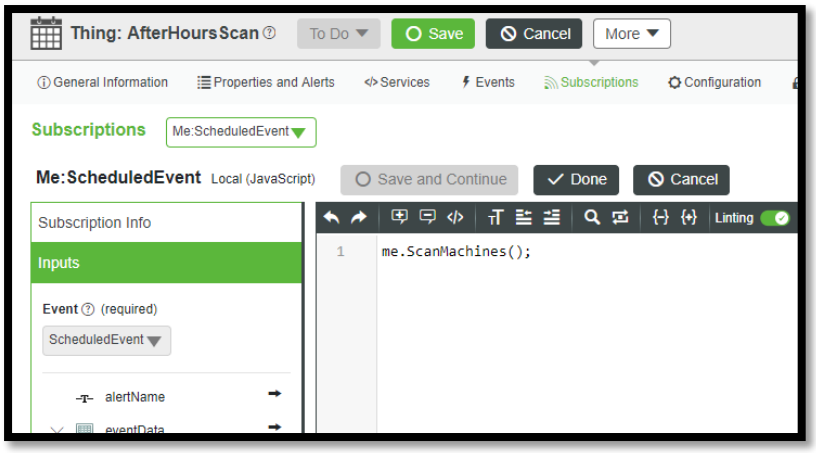


Figure 3-28: Subscription Code and Parameters

Figure 3-29 shows the code that was used in order to send an email message and an SMS text message (from a Twilio trial account) to the specified email and phone number. It further checks the properties of the _AdvancedTags—Mill1MachineState tag to see if it is true or not. If it is (the machine is on) then the service will be triggered and will send the email.

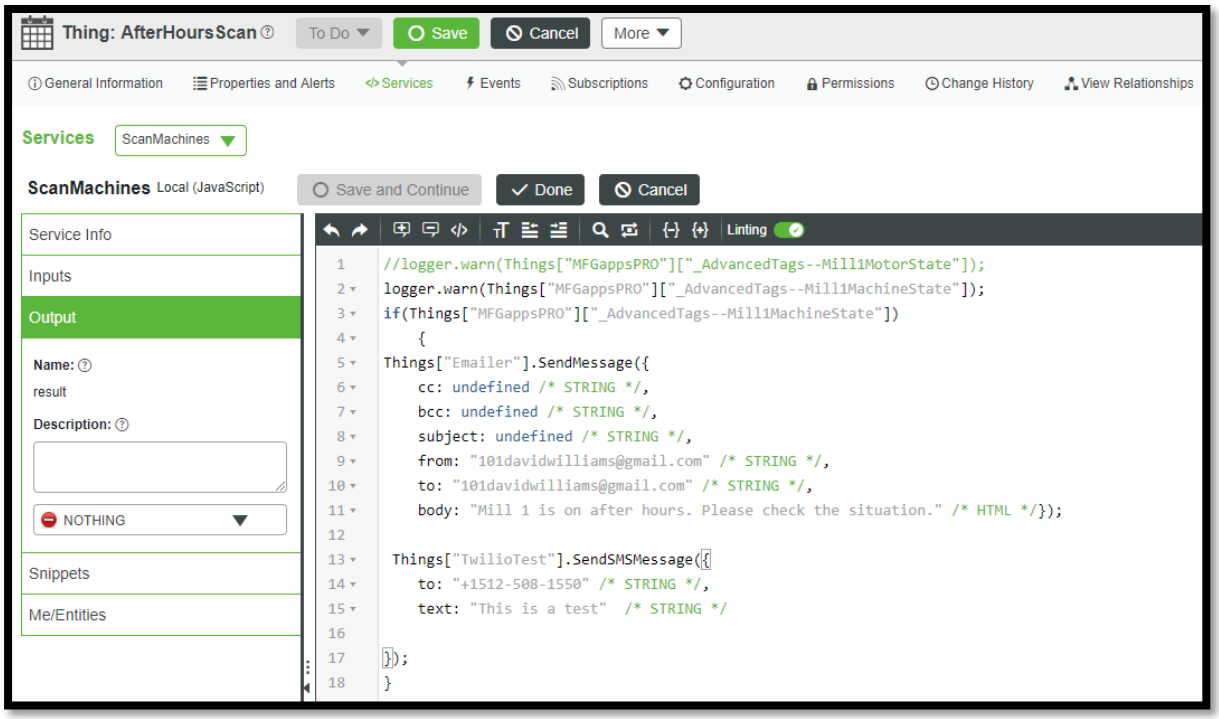


Figure 3-29: Service Code and Parameters

The EnableAfterHoursScan and DisableAfterHoursScan Things are then able to specify the times within which the AfterHoursScan can and will be triggered. The same process is followed: configure the time at which the subscription will be fired, create a service that then calls the EnableScheduler or DisableScheduler service respectively within the AfterHoursScan, then call the newly created service within its corresponding subscription.

3.4.2.2 Crash Notification

In order to send an alert to a supervisor of a crash, it was necessary to simulate what a crash would look like voltage-wise. To do this, a cylindrical aluminum piece of stock was placed into the chuck on Lathe_11 and 3 tests were performed, with the RPM of the lathe set to 950. The procedure of the test was to turn the spindle on, take a heavy depth of cut of more than 0.100" into the part in the Z-direction, and then ram the cross-slide into the part in the X-direction. After this was done, the brake was pressed to cut off the power to the spindle. The LabVIEW program was set up to monitor the voltage values during this experiment. After the experiments were performed, the data was analyzed and code was written in ThingWorx Composer to send an alert of the machine crash. This code consisted of setting a threshold range to define the normal voltage range of operation for the machine and spindle being on. From talking with Clint, the shop supervisor, a two-volt reduction was determined to be the threshold (anything below 161 volts with these machine state conditions) and would trigger an automated email to be sent to him and the TAs for the lab.

The reason that these experiments were set up this way was because it simulated some of the most common crashes that occur in that lathe machining environment. New students occasionally take too heavy of a cut, they turn the wrong hand-wheel, or leave a key in the chuck as they turn the spindle on. The experiment performed was set up this way to simulate a harder force of crash. Leaving a key in the chuck was not performed on this lathe as an experiment due to safety and because of the unpredictable trajectory of the key as it is flung out of the chuck. However, because of the sudden nature of a key crash and the material of the key being made out of a harder and more sturdy material than the aluminum stock used in the previously described

experiment, a safe level of assurance was reached that any key crashes would surpass the threshold determined and would still subsequently trigger the automated alert service.

3.4.3 Mills

After-hour operation and crash notification were the alerts that were requested by the shop supervisor to be set up and implemented into the smart system. Following are more details about the construction of the each of these alerts.

3.4.3.1 After-Hour Operation

The after-hour operation alert was enabled in a two-step process very similar to that of the lathe: set up a User inside the manufacturing apps with email and texting credentials and capabilities, and create a scheduler thing inside ThingWorx Composer that will scan the machines with a service to see if they are on or not at a specified time and have that service send an alert to designated user if the machine scan returns a positive result.

Creating and setting up users with email and text credentials and capabilities has already been discussed, but the ThingWorx step has some unique things that needed to be configured. Like the lathe, three schedulers were created to handle the machine scan that was to take place after hours in the machine shop: the AfterHoursScan thing, the EnableAfterHoursScan thing, and the DisableAfterHoursScan thing. Having these three things made it possible to only fire the service at a certain time of day/night. Figure 3-30 shows a snipshot of the code needed to send an automated alert for Mill 1. These three Things are able to be configured inside of ThingWorx Composer.

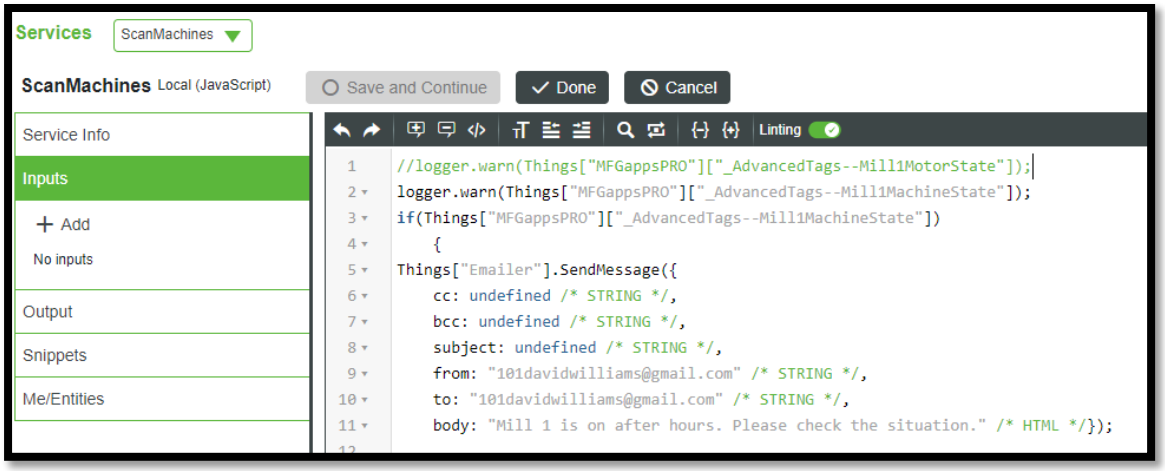


Figure 3-30: Code for AfterHoursScan Service

3.4.3.2 Crash Notification

In the case of the mills, the way to determine the crash notification was also to simply simulate a heavy cut or even a hard drive into the material with the tool. The procedure was to set up a cylindrical stock of aluminum into the vise, then take an exaggerated depth and force of cut into the part. The Quick Client from KEPServerEX was used to help monitor in real time the effects of the crash on the different pieces of information being tracked by the ETH-200 communication card. The thresholds and code would then be set and written to trigger an automated alert to the specified persons.

4 RESEARCH RESULTS AND ANALYSIS

The results and analysis of achieving unified connectivity, real-time monitoring, and issue identification are given in this chapter. Results and analysis for the lathe and the mills will be discussed separately. The following table lists the overall results of achieving the three fundamental smart components for the lathe and four mills and the following sections will discuss these results in detail.

4.1 Unified Connectivity

This section discusses in detail the results that were obtained for unified connectivity for both the lathe and the mills. This will cover any iterations made in establishing the final connection between the disparate devices to the ThingWorx Manufacturing Apps, as well as any qualifying factors obtained from data analysis.

4.1.1 Lathe

Table 4-1 provides a summary of the two different methods investigated analyzed in order to determine the best solution for lathe unified connectivity. Ultimately, the OPC UA connection was selected to be the method utilized in the implementation of connectivity for this factory. Although both methods were successful in establishing a connection, the OPC UA method was clearly more advantageous.

Table 4-1: Lathe Unified Connectivity Summary and Analysis

Type of Connectivity	Speed (1 push of data)	Reliability of Connection	Design Intent	Other
REST API	30 ms	Experiences "hiccups" in connection, causing stop of program	Built for simple send/receive communication between 2 devices (one on one)	n/a
OPC UA	29 µs	Configured to reset upon loss of network connection	Designed to unify disparate devices and protocols for industrial setting	Specific driver in KEPServerEX designed for this type of connection, allowing direct relay of data to the manufacturing apps

Lathe unified connectivity was achieved by establishing an OPC UA connection between LabVIEW and KEPServerEX. The detailed LabVIEW program outlining how this was done is given in Appendix A: LabVIEW Code. Figure 4-1 shows a screenshot of the Quick Client inside KEPServerEX that shows Good quality data connections with values being updated.

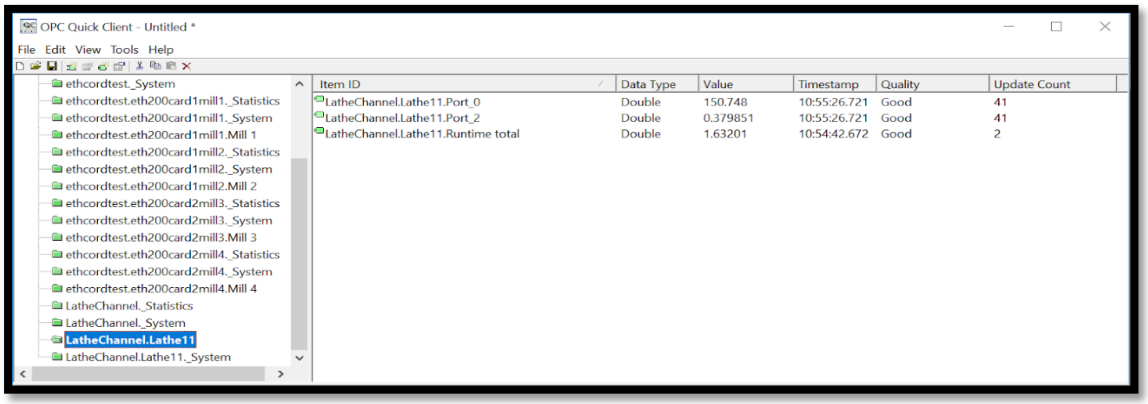


Figure 4-1: Working Lathe Connectivity

Figure 4-2 shows the Quick Client when there is a Bad quality connection for the tags. This occurs when the LabVIEW program is ended or goes offline. As shown, the last known values are utilized and displayed in KEPServerEX, and these are the values that will be shown in the manufacturing apps.

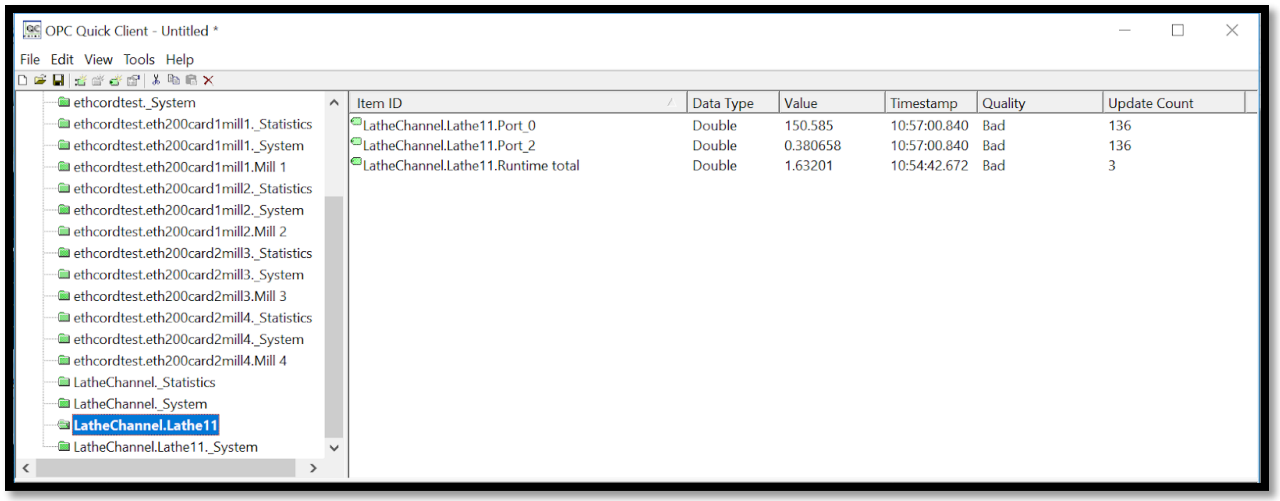


Figure 4-2: Bad Lathe Connectivity from Stopped LabVIEW Program

Because the last known values are used at the time the connection goes bad, it may be difficult to tell if the machine is actually producing good data. There needed to be a way to track whether there was a good connection or not between the physical device and the digital representation of that device inside of KEPServerEX. After further exploration of the Quick Client in KEPServerEX, it was discovered that there are tags automatically generated upon creation of the device's tags and the execution of the Quick Client service. Figure 4-3 shows this helpful tag: the System_Error tag. When this Boolean-based tag has a true value (1), this signifies that there is an error in the connection. When it has a false value (0), then the connections for all of the rest of the tags have a Good quality. This tag, along with user-created

tags, may be sent up to the manufacturing apps and utilized for monitoring and issue identification purposes.

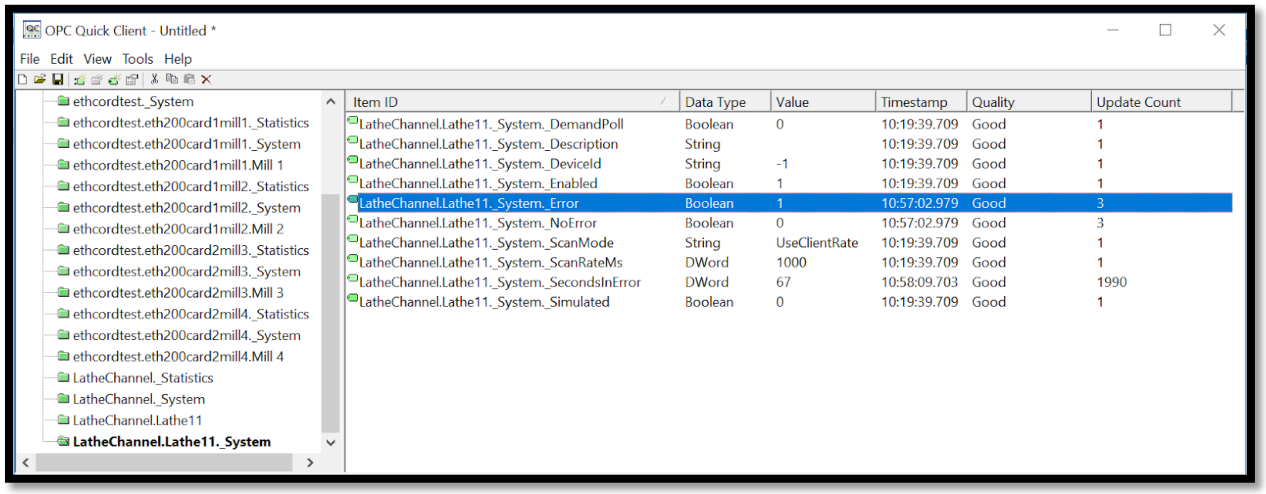


Figure 4-3: System_Error Tag Signifying Communication Loss to the Device

4.1.2 Discussion of Results

4.1.2.1 Iterations Taken to Get to Final Version of LabVIEW Program

There were several versions to get to the final iteration of the LabVIEW program being used to establish unified connectivity for the lathe in this project. The first version, shown in Appendix A: LabVIEW Code, has LabVIEW sending its data via REST API to directly change the property values of a Thing inside of ThingWorx. Although this method worked, it was not as suitable a solution as was the OPC UA connection to KEPServerEX. There are several reasons for this.

One of the biggest reasons for choosing to go with the OPC UA server connection is that this method is simply faster and does not rely on the network for pushing its data like the REST

API version does. After setting up an experiment to measure the data rates between these applications, it was discovered that the OPC UA connection took 29 microseconds in LabVIEW to send its data to KEPServerEX, whereas it took the REST API version several milliseconds. After consideration, this makes sense because REST protocol was simply not built or designed to be sending or receiving massive amounts of data in an industrial setting like the OPC UA connection was. Additionally, the REST API version would be limited as to the number of machines that it would be able to acquire data from and send data to. The OPC UA connection was meant to be able to connect to other machines that had other protocols and bring those data streams into one flow.

Another reason that the OPC UA connection method was used is because of its connection to KEPServerEX, which the manufacturing apps were intended to be used with. If the REST API version was pursued, then mashups and customized code would be required for every piece of data for every machine which is what the apps were designed to reduce and, if possible, eliminate. Using the REST API version frankly would require a lot of training, time, and would not be an effective use of resources for the shop supervisor, or any other engineer out in the field that may need to do this for dozens or even hundreds of devices.

4.1.2.2 24/7 Connectivity

A minor problem that was encountered with initial setups was the fact that with a demo version of KEPServerEX came a limitation on how long the program would keep track of data. In the demo-version, a two-hour time limit was imposed. This would not be an acceptable solution in an industrial environment, so a more permanent option was needed. A professional license of KEPServerEX 6.3 was obtained and activated to allow 24/7 connectivity with the

factory devices. Instructions on how to manage and activate a license for KEPServerEX can be found at my.kepware.com.

4.1.2.3 Common Causes of Bad Connections

One of the most common causes of a "bad" quality connection in the manufacturing apps is that the LabVIEW program was ended or aborted manually, or if there was a disruption in connection between the cDAQ chassis and the network. When this happens, the connection of the program to the device is severed and will naturally cause a disrupted connection to KEPServerEX and change the quality of the data connection from "good" to "bad", which will then also display a bad quality piece of data in the manufacturing apps. That is why it was important to set up a program that would run reset itself in the event of a hiccup in network connection to the chassis. By resetting itself (which takes a matter of seconds), connectivity is preserved, and the program will not fault out.

Other factors may need to be adjusted based off of application, such as the rate at which data is being acquired/sent, for example. If the rates are too dissimilar, then the program may cease to function as well. This applies to each different DAQ device, but once a reliable frequency is set (1000 Hz for this project), the device functions reliably.

4.1.2.4 Limitations

There are some limitations to this method of establishing unified connectivity with an older machine. One of the first and foremost is the cost associated with this setup. It was approximately $500 for the chassis alone. Because of the large amounts of voltage going through the lathe that we needed to tap into, the module had to be able to accommodate for large voltage

values. The NI-9225 voltage module used for this device was $1500. Additionally, the module only had three screw-in voltage connections that could be utilized, limiting the number of pieces of data that could be acquired from the lathe. There are at least 10 other lathes in this factory alone. It would be difficult to justify such an investment using this solution. National Instruments was chosen for its exceptional ability to take measurements and for its universal use in the manufacturing industry. Costs on this setup may be a limiting factor, but the principles are sound and steps for implementation would prove beneficial for other types of systems already using National Instruments products.

4.1.3 Summary

Is it possible to establish unified connectivity between manual lathes/mills and a computer?

From the setup and results of the lathe portion of this project, it is possible to establish unified connectivity between a manual lathe and a computer. The lathe connectivity was accomplished by measuring and acquiring data using National Instruments devices and having its control program, LabVIEW, send that data to a computer to a commonly used piece of software in the manufacturing industry called kepware (KEPServerEX) via an OPC UA connection. 24/7 connectivity was set up with the ability to send its data to a visualization tool on an IIoT platform called the ThingWorx Manufacturing Apps. Although setup was costly, the principles and steps used prove valuable for implementation of other types of NI products across the manufacturing industry everywhere.

4.1.4 Mills

Table 4-2 provides a summary of the connectivity methods investigated in achieving unified connectivity in the mills of the factory.

Table 4-2: Mill Unified Connectivity Summary and Analysis

Type of Connectivity	Speed (1 push of data)	Reliability of Connection	Reason for failure/success	Other
Direct connection between drive and computer (Ethernet cord)	40 ms	No communication established	Older version of Toshiba protocol trying to communicate to modernized KEPServer driver	Difficult to say whether the Toshiba suite would cover the older version of this protocol
Modbus RTU (RS232 cord from ETH card, null modem adapter, RS232 to USB converter)	40 ms	No communication established	Same as above	Older type of connection, less reliable for communication
Modbus RTU (Ethernet cord from Modbus TCP/IP port into Ethernet port on computer)	40 ms	Communication established (first configured on embedded webserver, then streamed directly into KEPServerEX)	Successfully converted older Toshiba protocol into a more modern and used protocol	KEPServer driver available (tag addresses start in the 40000 range)

Connectivity of the four mills in this project was achieved by converting the Toshiba protocol being tracked on the mill inverter into Modbus TCP/IP with an ETH-200 communication card. Once converted into a protocol that KEPServerEX could communicate

with, it was a straightforward process relaying that information to the manufacturing apps. Figure 4-4 shows the Good quality connections made on one of the mills.

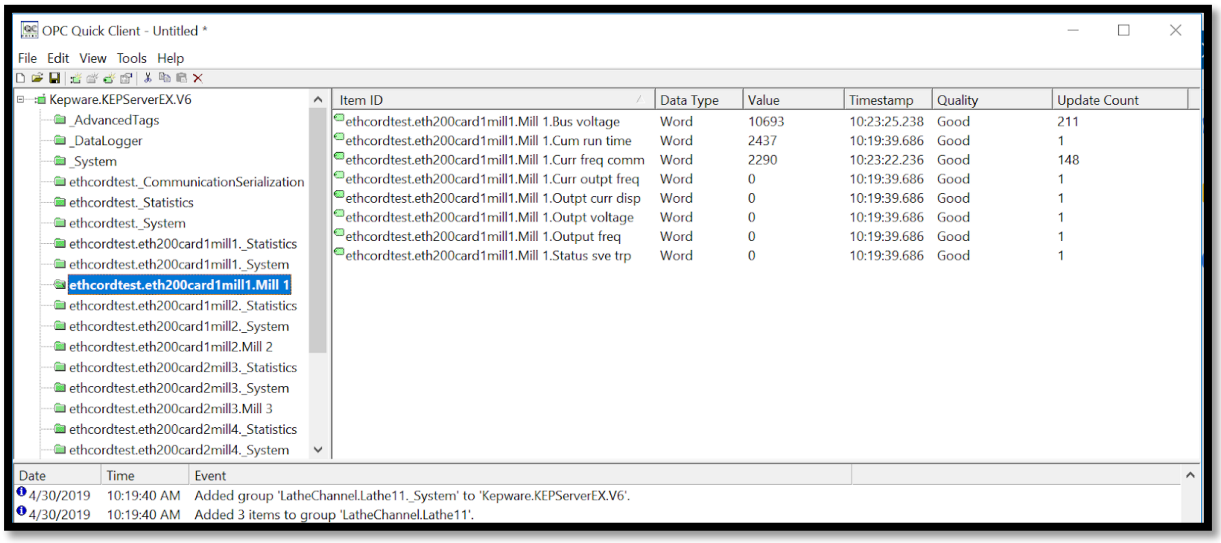


Figure 4-4: Mill 1 Connectivity to KEPServerEX

Somewhat different than the lathe connection, the only time when the connection goes bad (assuming that the wiring and hardware are all functioning properly) is when the machine goes into an e-stop state. When this happens, the last Good values are retained as the tag values in both KEPServerEX (and subsequently the manufacturing apps) as shown in Figure 4-5.

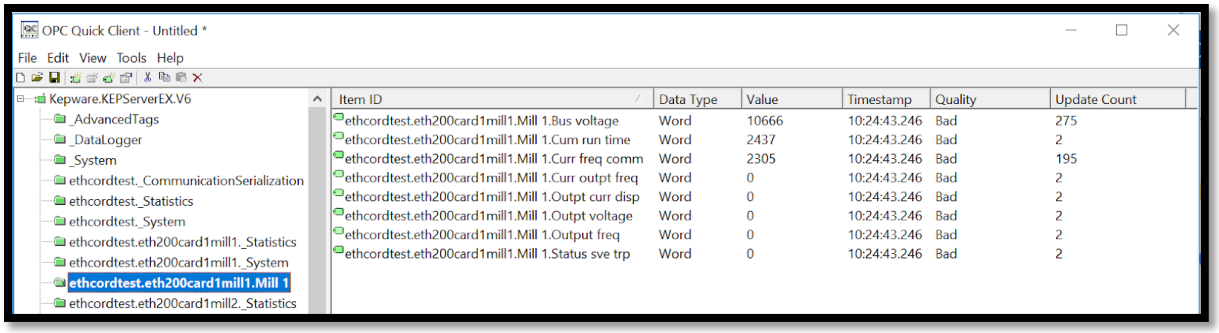


Figure 4-5: Quick Client of Mill 1 when E-Stopped

Subsequently, the Sytem_Error tag changes to true, allowing the apps to recognize that the machine is simply in an e-stop mode. This tag tracks the condition of the connection (Figure 4-6). If the value remains true on the System_Error tag even after the machine is pulled out of e-stop, then the connection may need to be checked.

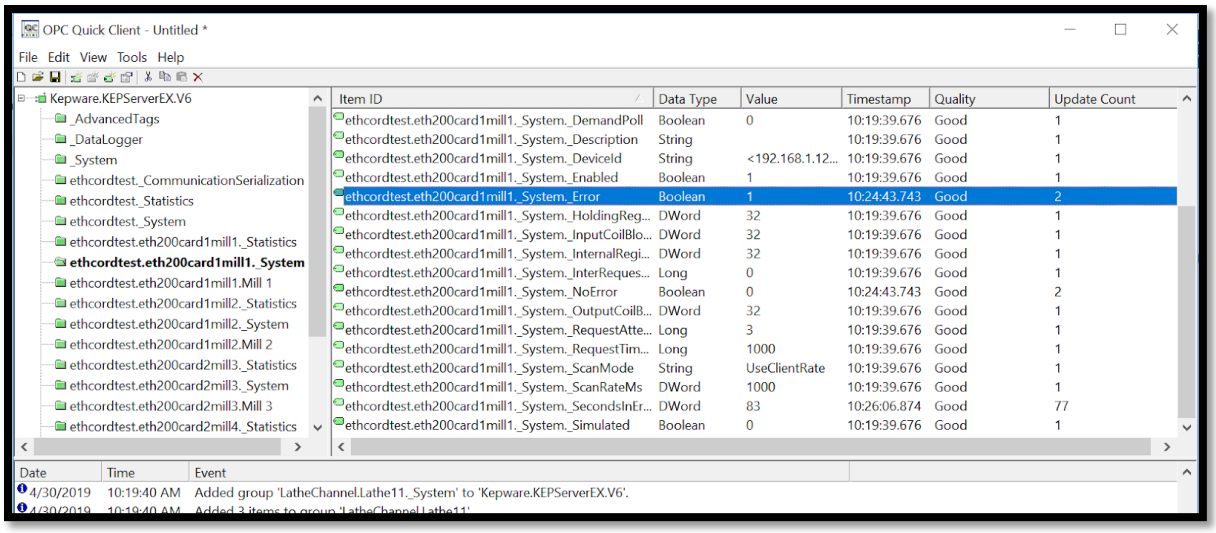


Figure 4-6: Indicator of Mill 1 Communication Loss

4.1.5 Discussion of Results

4.1.5.1 Iterations of the Connectivity Setup

In the initial attempt to establish communication between the drive and the computer, an Ethernet cord was connected into the serial port on the drive and the Ethernet port on the computer. When this happened, no communication occurred between KEPServerEX (on the computer) and the drive. After thought and consideration, it was determined that the correct protocol and driver combination needed to happen so that KEPServerEX would know how the data would be formatted coming in, and be able to respond and process that data accordingly.

This led into an investigation of the protocol being used on the drive. The protocol that the drive was discovered to be using is an older version of Toshiba protocol.

In the second attempt of connecting a mill to a computer, an attempt to have the Toshiba protocol be converted into Modbus RTU was made. Cords and hardware were ordered to pursue this type of connection and the Modbus driver would be utilized inside of KEPServerEX. Figure 4-7 shows this connection. This setup uses RS232 cords to connect the mill to the card and then to the computer.

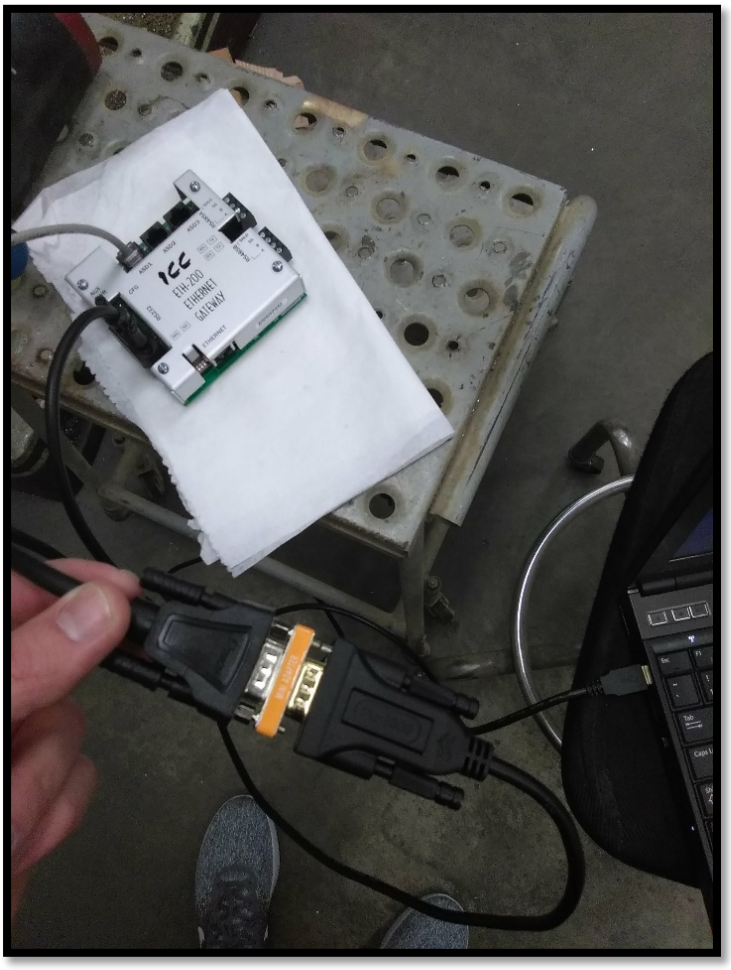


Figure 4-7: Connectivity Attempt Using RS232 Cords/Modbus RTU Protocol

However, after additional thinking, in the industrial setting, it would be unrealistic to have companies go and get an outdated cord and keep the language in an older protocol. Consequently, it was decided to attempt using the Ethernet/IP port and protocol on the communication card using a standards Cat 5 Ethernet cord. After configuring the channel, device, and tags/addresses, the data came through to the computer and connectivity was achieved, using a much more modern and ubiquitous protocol. Now the pieces of information on the inverter already being tracked could be sent to the apps, without any complex code needing to be written.

4.1.5.2 E-Stop Problem

After one mill was successfully communicating to KEPServerEX, Mill 2 and Mill 3 were added to the ASD2 and ASD3 ports on the ETH-200 card respectively. The problem that arose from this, however, was that on occasion, when one mill was e-stopped, the other tags from the other mills would also cease being passed to KEPServerEX. The pattern in which they would fail was unpredictable. Up to this point in the setup, the communication card was being powered by the inverters on the mills themselves, and it was theorized when the card stopped receiving power from its original source inverter, communication was disrupted for the entire card. Consequently, auxiliary power cords were ordered to supply constant electricity to the card, and a second communication card was ordered to have two mills on each card. When these things were done, the e-stop dilemma was resolved, and the machines did not affect the power supply of the card, providing consistent and reliable connectivity.

4.1.5.3 Cost of Setup

The cost of this set up was quite affordable. The cost for the communication card was approximately $500. This would allow constant communication of up to three mills per card. The other software needed would be KEPServerEX, which is a part of many companies already, and the manufacturing apps, which is a free downloadable extension of ThingWorx Composer.

4.1.5.4 Limitations

One of the main limitations of this connectivity setup lays in the scan rate of the communication card, which is dependent on the data rate of the inverter itself. After looking in the manual for the baud rate (9600) and conversing with an applications engineer, it was provided that the speed of data coming from the inverter to the communications card for 8 pieces of information takes approximately 40 milliseconds. For other applications in industry, this may prove to be a strength or a weakness in this solution. If the machine that is sending the information has a slow data send-rate, then the communication card will follow suit: the flow will only be as fast as the slowest piece in the setup.

4.1.6 Summary

Is it possible to establish unified connectivity between manual mills and a computer?

From the setup and procedures shown, it was discovered that it is indeed possible to establish unified connectivity between manual mills and a computer. With this setup, a quick and easy solution to establish unified connectivity was constructed to a device that was not initially meant to be tracked and monitored using a relatively inexpensive communication card to convert

Toshiba protocol into a more common Modbus TCP/IP protocol, which could then be used in conjunction with KEPServerEX.

4.2 Real-Time Monitoring

This section discusses in detail the results that were obtained for real-time monitoring for both the lathe and the mills. Due to the lathe and mills being electrically and mechanically different from the other, the pieces of information gathered were also different.

4.2.1 Lathe

Table 4-3 depicts that numerous pieces of information that were able to be obtained and monitored in real-time within the manufacturing apps. It also outlines some of the challenges that were encountered in the process, as well as any next steps that may be desired to be explored in future research.

Table 4-3: Lathe Real-Time Monitoring Summary and Analysis

Name	Description	Source Origin	Logic Written	Challenges Encountered	Next Step
Runtime_total	Cumulative runtime	LabVIEW	n/a	LabVIEW code writing	n/a
Port_2	Voltage Port_2	LabVIEW	n/a	n/a	n/a
Port_0	Voltage Port_0	LabVIEW	n/a	n/a	n/a
Machine State	Derived tag(s) KEPServerEX	KEPServerEX	In KEPServerEX opf file	Obtaining thresholds	Add more sensors
_Error	System tag in KEPServerEX	KEPServerEX	n/a	n/a	n/a
Running	Machine status	Manufacturing apps	(in apps)	n/a	n/a
Unplanned Downtime	Machine status	Manufacturing apps	(in apps)	n/a	n/a
Warning	Machine status	Manufacturing apps	(in apps)	n/a	n/a
Planned Downtime	Machine status	Manufacturing apps	(in apps)	n/a	n/a
Availability	Availability of machine	Manufacturing apps	Run Time / Planned Production Time	Inaccurate default formula	Change default formula
Quality	Quality of parts coming off of line	Manufacturing apps	Good Count / Total Count	Education setting (hard code solution)	Set up application for using this
Performance	Ratio of actual run rate to ideal run rate	Manufacturing apps	(Total Count / Run Time) / Ideal Run Rate	Education setting (hard code solution)	Set up app
OEE	Overall Equipment Effectiveness	Manufacturing apps	Avail* Perf* Qual	Inaccurate default formula	Change default formula

On the lathe, two out of the three ports were used to acquire voltage measurements. Figure 4-8 shows bound properties being pulled from KEPServerEX. From the raw voltage values being taken, the state of the machine was able to be determined. From two simple measurements, the operator can now see the most common machine states (machine on/off, motor on/off, etc.). These were the main components that were important for the initial setup of real-time monitoring.

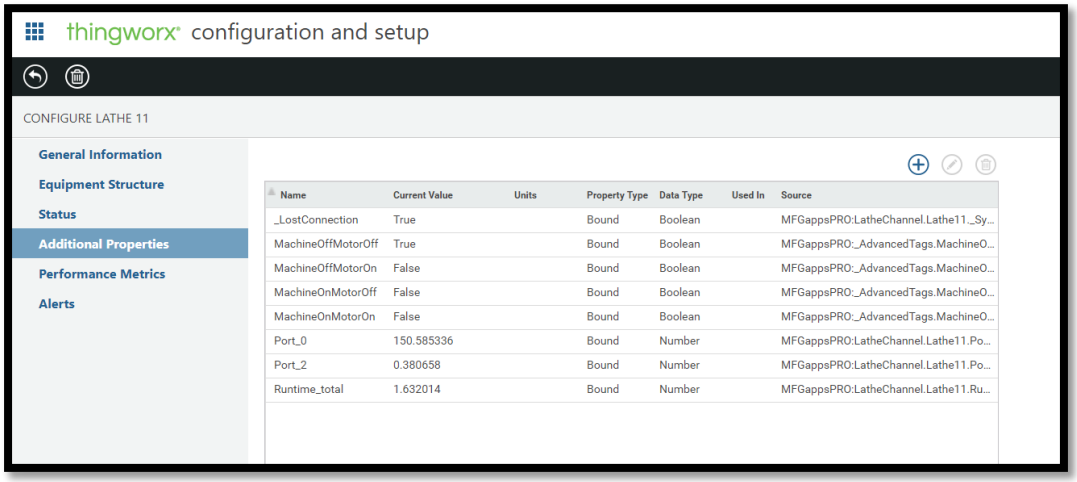


Figure 4-8: Additional Properties Tied to KEPServerEX Tags

Figure 4-9 shows the view of the lathe available under the Asset Advisor tab of the manufacturing apps. It shows the display state, as well as how long it has been in that display state, if there are any active alerts, and the number of alerts that have occurred that week. These different pieces of information may be modified and configured differently if desired. Instructions on how to do this are located in the Customization Guide for the ThingWorx manufacturing apps.

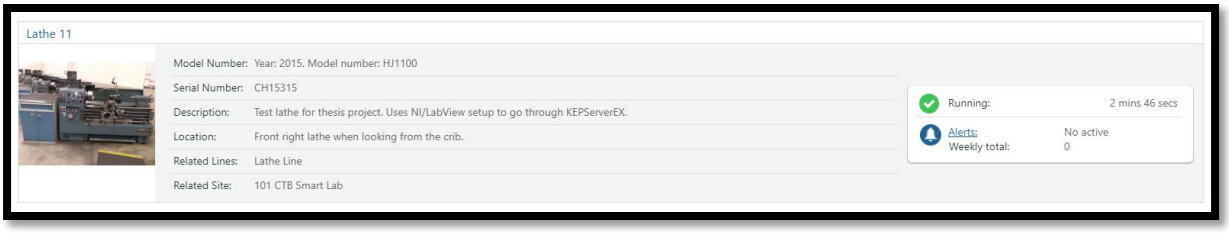


Figure 4-9: Asset Advisor View of Lathe_11

Figure 4- 10 shows the status definitions for Lathe 11. With the manufacturing apps, the default statuses that can be defined are Running, Unplanned Downtime, Warning, and Planned Downtime. The Running status is defined to be when the spindle is on. Unplanned downtime is defined to be when the machine and motor are off but the machine is not in scheduled/planned downtime. Warning is the status whenever one of the alerts for the lathe/asset is active. Planned downtime is user defined as the time of day or week in which the machine is scheduled to be worked on or maintained. These statuses were based off of the values coming in from KEPServerEX and are defined in the Expression sections, as shown. The status updates approximately every 30 seconds. After researching in the configuration manual, it was discovered that other features, such as KPIs have a default update rate of once per minute. The convenient feature about the manufacturing apps is that they are completely configurable and may be changed based off of the needs of the application or environment. The fastest update rate that it will allow the user to set is 1 minute for KPI evaluation, but the status updates approximately every 30 seconds. These definitions are based off of the dynamic tags being pulled in from KEPServerEX which are then tied to the lathe. Although somewhat limited in the quantity of information being pulled, enough information was pulled to be able to define these numerous machine statuses.

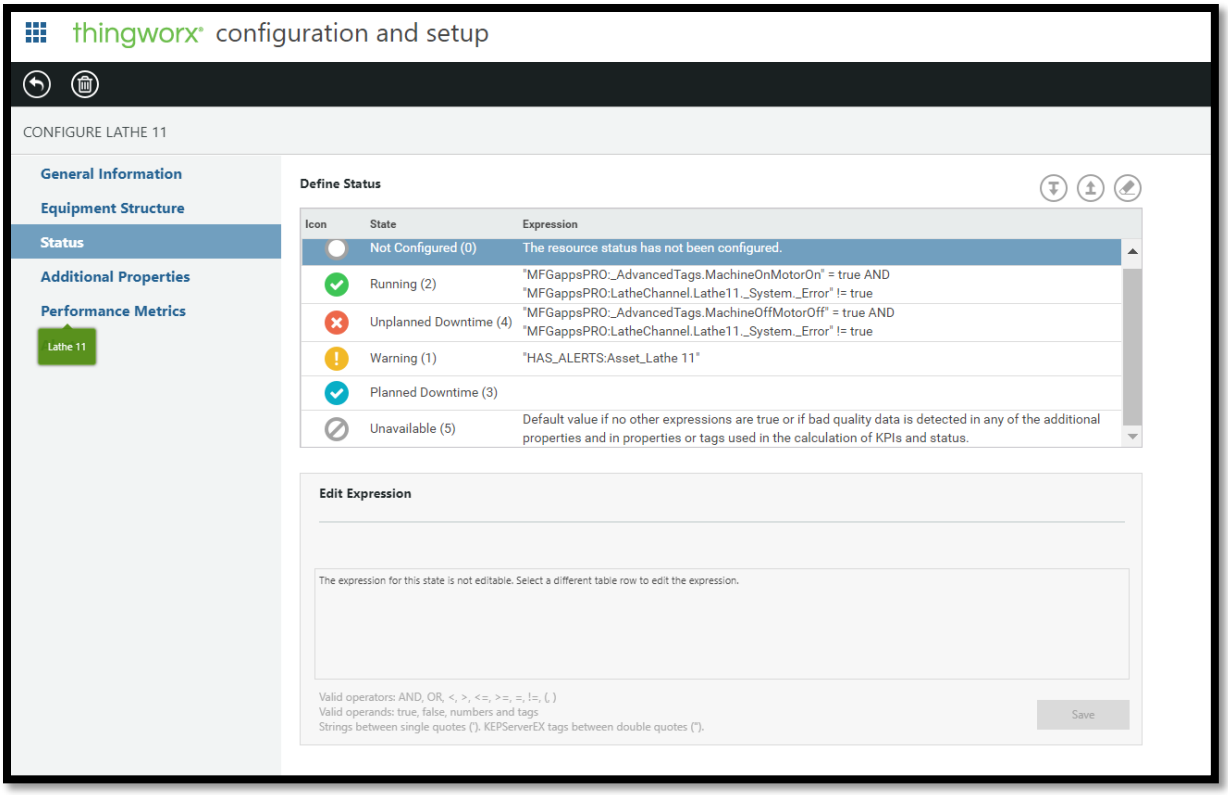


Figure 4- 10: Status Definitions for Lathe 11

Figure 4-11 shows the production key performance indicators (KPIs) of the machines within the factory. It is set up to track overall equipment effectiveness (OEE), availability, quality, and performance. Additionally, it shows the story of the data over time for that KPI calculation period. In this case, the calculation period is 5 minutes. This KPI calculation time period may be adjusted (1-99,999 minutes). As mentioned earlier, this update is configurable within ThingWorx Composer. Instructions on how to do this are located in the Customization Guide. This will guide walks through how to set up all of the Things and Services that control what is shown and displayed inside of the manufacturing apps. This type of configuration should be done by an admin for the apps, not the TAs in the lab or the shop supervisor.

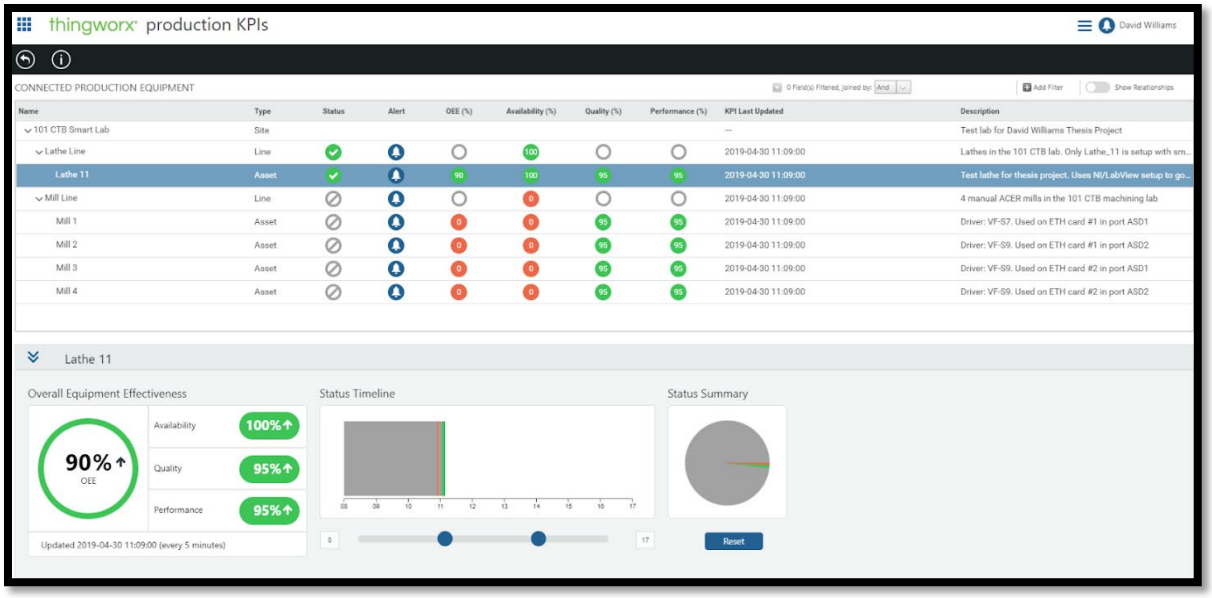


Figure 4-11: Production KPIs View of Lathe_11

4.2.2 Discussion of Results

4.2.2.1 Advantages

There were many advantages in the way that real-time monitoring was set up in this project. The user-friendly display visuals offered insight into the machines status and performance, as well a history of those factors. The manufacturing apps come with code already built into them. This is useful because now the engineer or supervisor does not have to learn new code script to show the data that he/she desires. It is already customized for the manufacturing setting, and is customizable as well.

4.2.2.2 Limitations

The main limitation in the monitoring of the lathe was the variety of data that was able to be monitored. Because only two ports on the cDAQ were used, a limited amount of information

could be reliably obtained. A third port could have been used, but would have yielded little more results. The initial thought was to use two ports, that would then determine the states of the machine, motor, and brake. Table 4-4 shows the different machine conditions in a matrix format and served as the template for the following tests that were performed.

Table 4-4: Machine Conditions Matrix

Test	Factors (+/-)		
	On/off switch	Motor switch (on/neutral)	Brake (pressed/not pressed)
1	-	-	-
2	+	-	-
3	-	+	-
4	+	+	-
5	-	-	+
6	+	-	+
7	-	+	+
8	+	+	+

The following figures represent the eight tests that were performed and the voltage behaviors under each of the main machine conditions. These eight tests represent the conditions represented in the table above. These conditions were determined based off of the most commonly used conditions to which the machine would be subjected. Other factors could be included in future tests (coolant status, overhead light status, etc.) but were not deemed essential in desired data for real-time monitoring. The desired pieces of data were determined by the shop supervisor and are represented in Table 1-1. Discussion of the results in monitoring these pieces of data is given later on in this chapter.

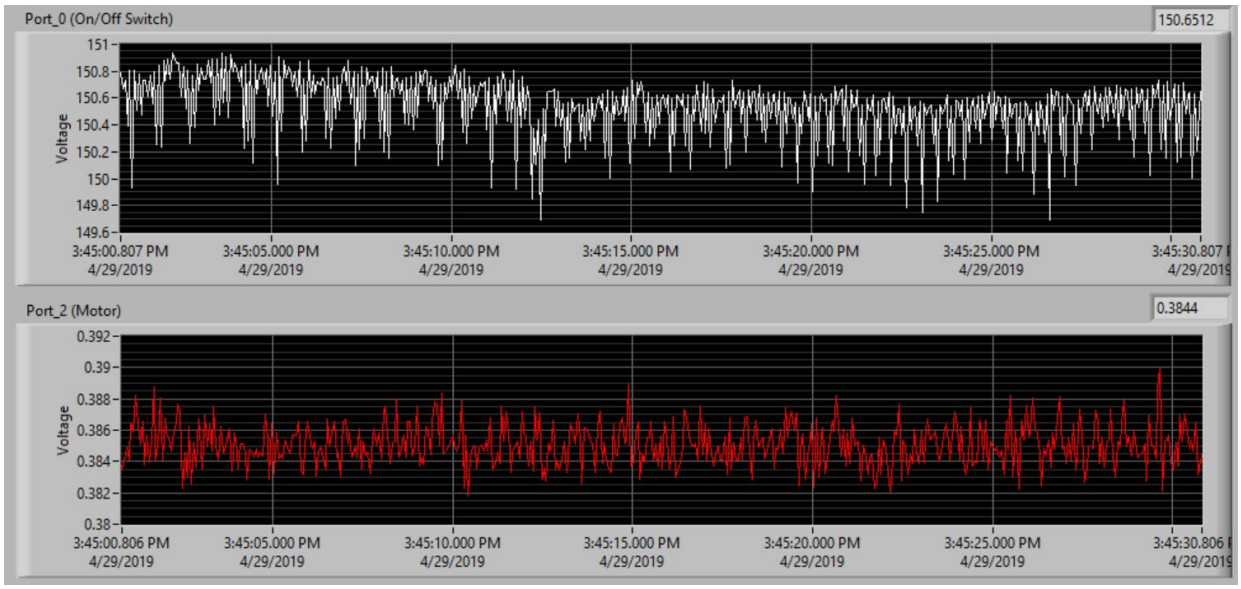


Figure 4-12: Lathe Test 1-Machine Off, Motor Off, Brake Not Pressed

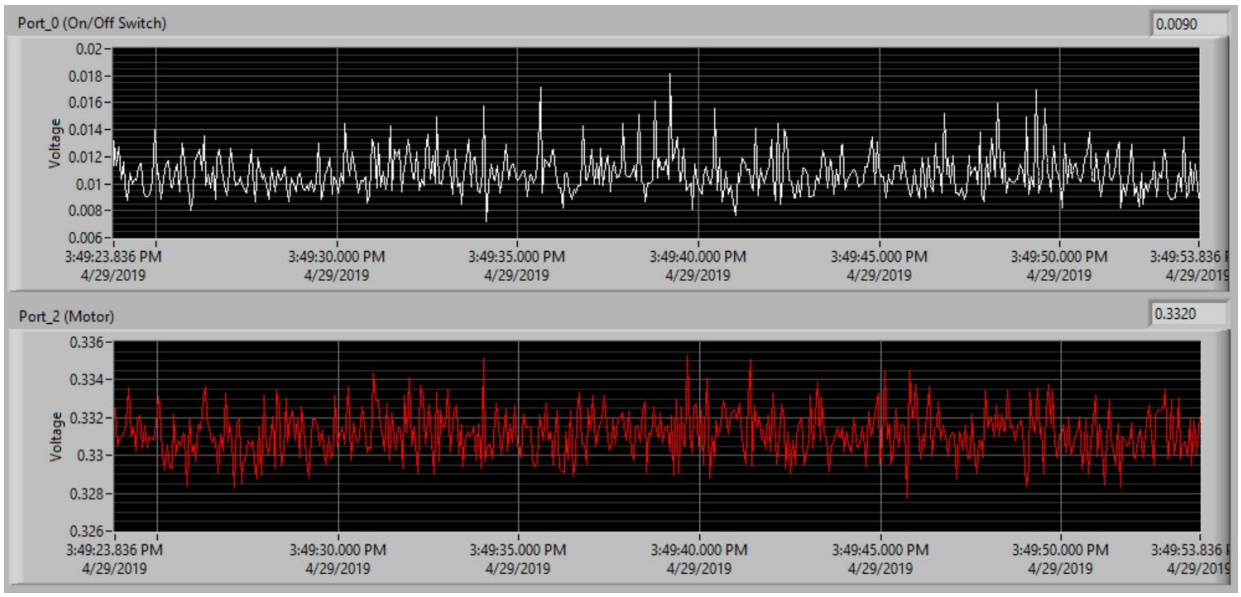


Figure 4-13: Test 2-Machine On, Motor Off, Brake Not Pressed

Test 1 demonstrates the machine voltage conditions when the machine is considered turned off. Test 2 represents the voltage behavior when the on/off selector is switched to ON.

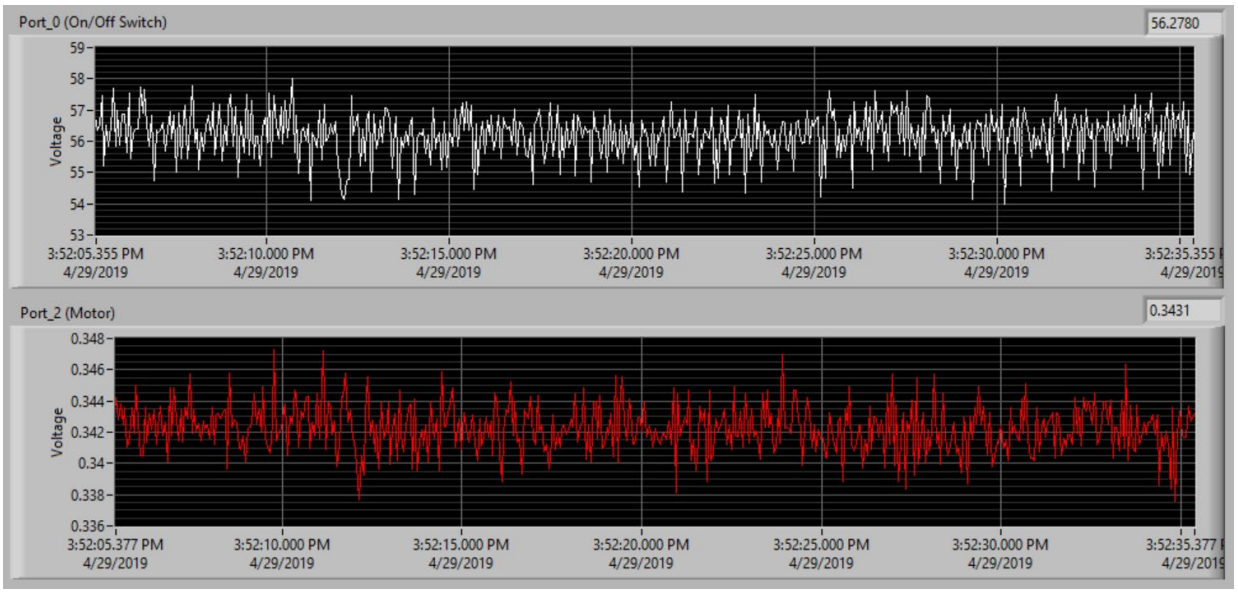


Figure 4-14: Lathe Test 3-Machine Off, Motor On, Brake Not Pressed

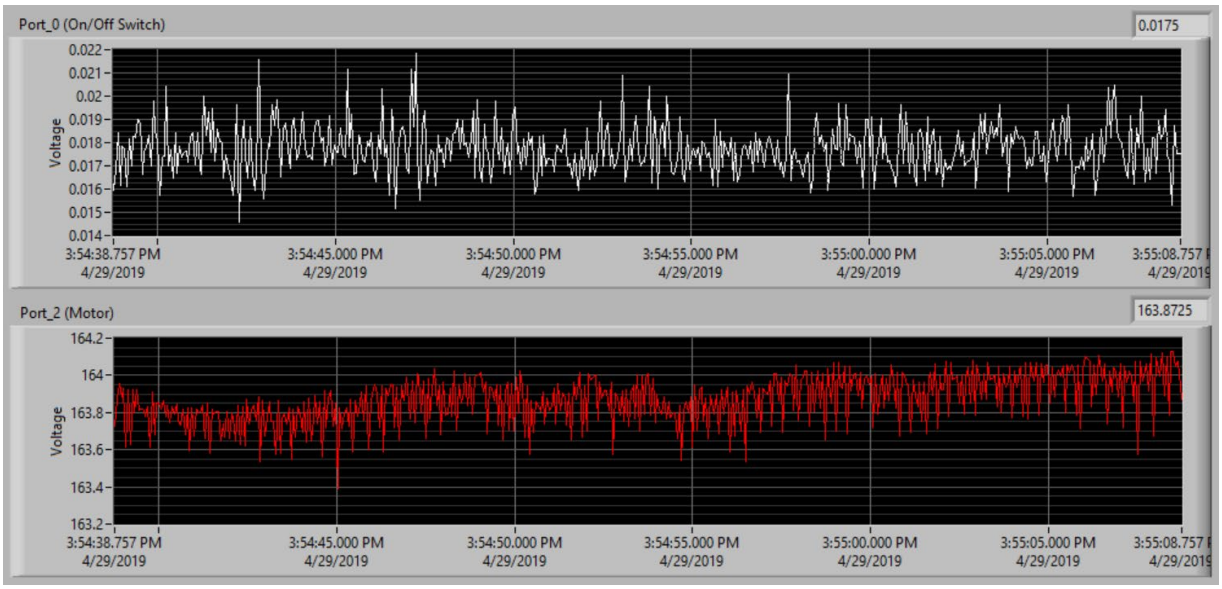


Figure 4-15: Lathe Test 4-Machine On, Motor On, Brake Not Pressed

Test 3 depicts what the voltage would be like if the machine was switched off but the motor lever was still engaged. Test 4 shows the normal operating voltages of the machine being on and the motor lever is engaged. This is consistent despite varying RPM values.

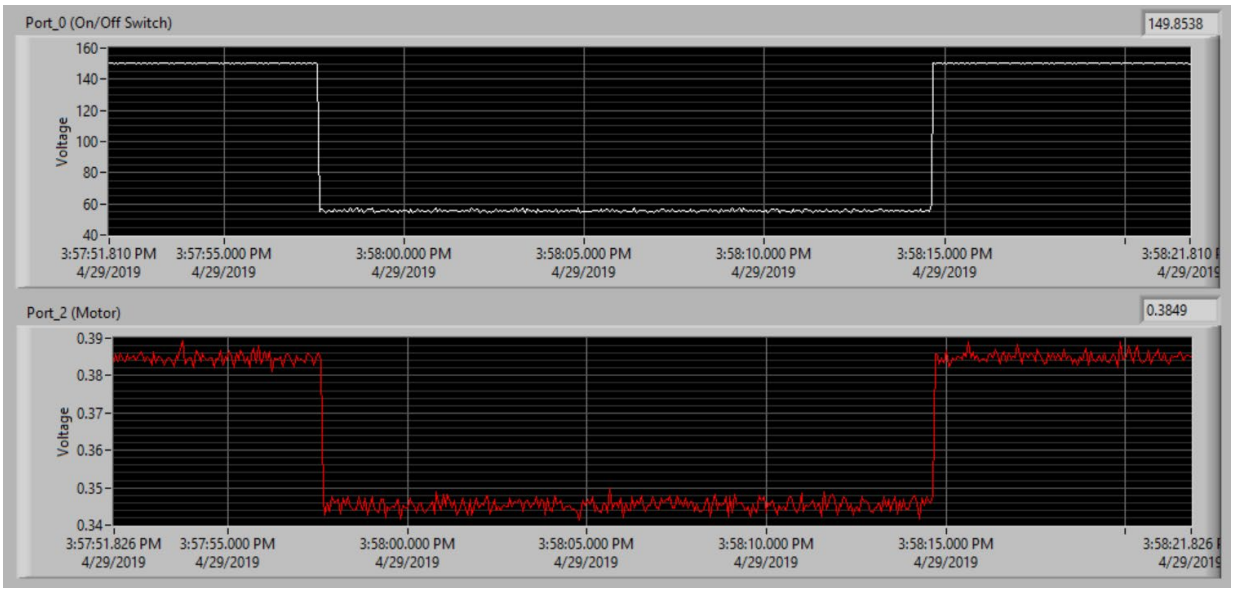


Figure 4-16: Lathe Test 5-Machine Off, Motor Off, Brake Pressed

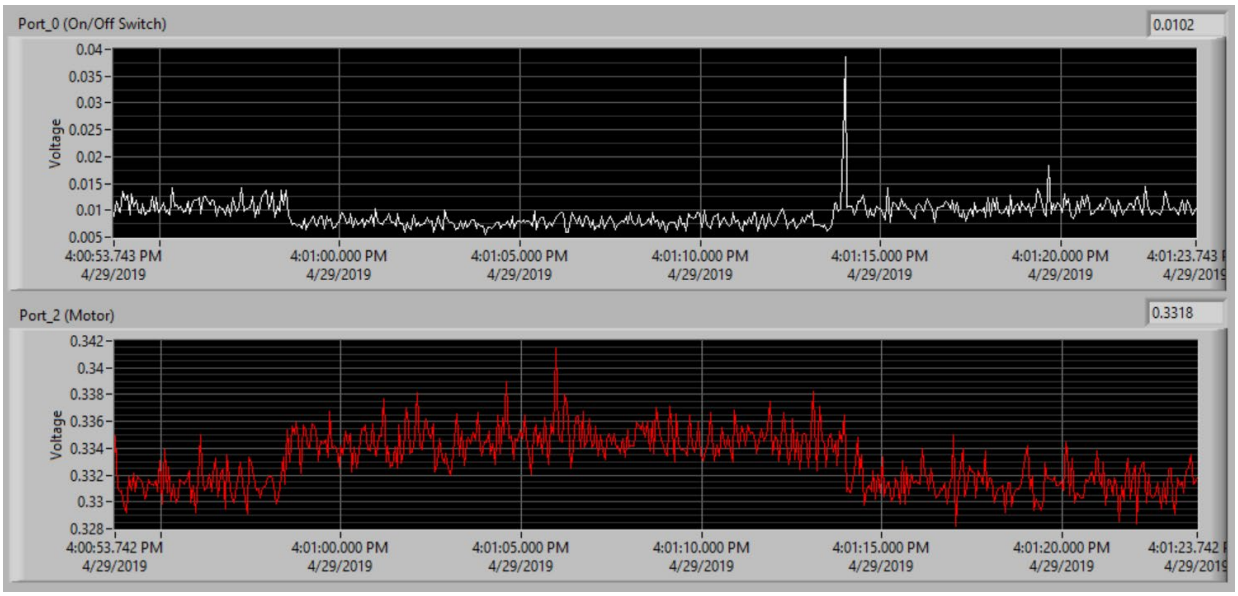


Figure 4-17: Lathe Test 6-Machine On, Motor Off, Brake Pressed

Test 5 introduces the brake into the equation and shows what the behavior is when the machine and motor are off, but the brake is pressed. Test 6 has the turned on but the motor is still turned off.

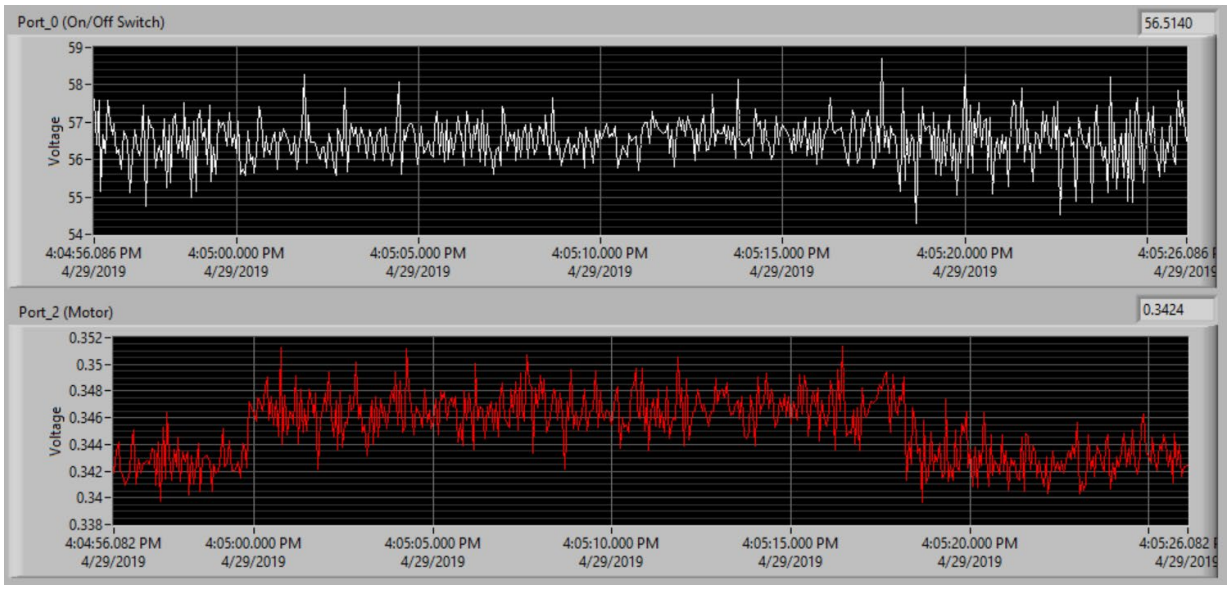


Figure 4-18: Lathe Test 7-Machine Off, Motor On, Brake Pressed

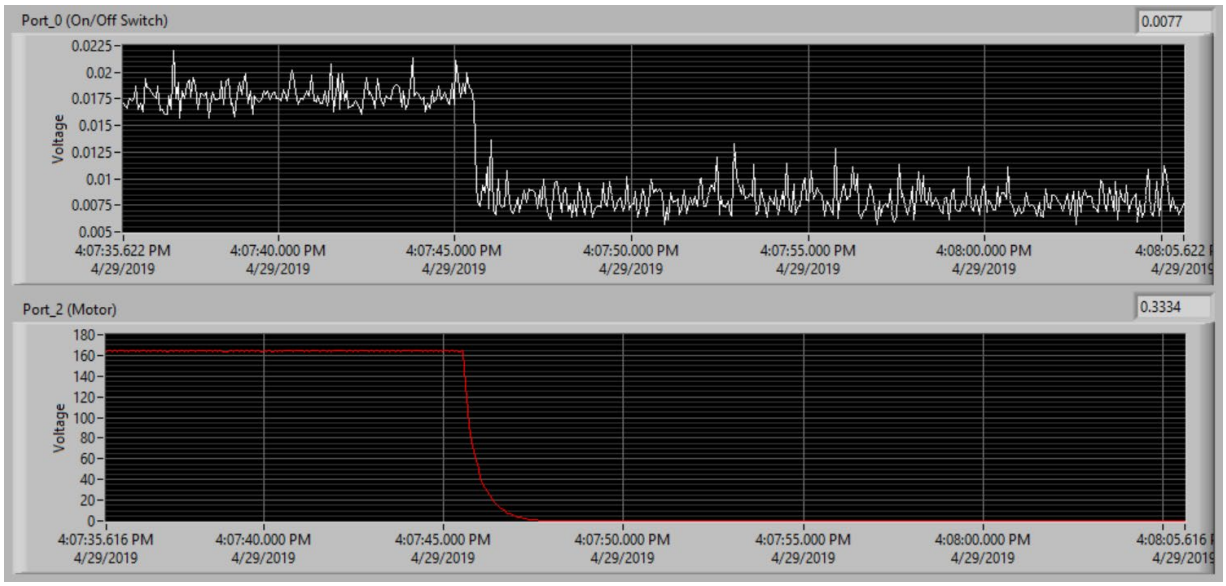


Figure 4-19: Lathe Test 8-Machine On, Motor On, Brake Pressed

Test 7 represents the voltage behaviors under the machine being on and the brake is pressed. Test 8 shows the behavior when the machine and spindle are running, but then the brake is pressed.

For the on/off switch and the motor selector aspects, the voltages were very distinct and provided evidence that these two channels would be able to have reliable real-time monitoring. However, problems arose after trying to monitor aspects of the machine that did not directly relate to the ports being measured. The on/off switch channels and the motor spindle channels were tapped, but trying to reliably distinguish when the brake was being applied proved unreasonable. Because the brake being applied did not affect the values of the voltage on the two ports being measured significantly, a reliable monitor on the brake could not be obtained. A sample test that emphasizes this unreliability to distinguish additional features is shown below. Figure 4-20 shows the voltage behavior of Port 0 and Port 2 as the machine is on with the motor running, and then the motor lever is switched to the off-position.

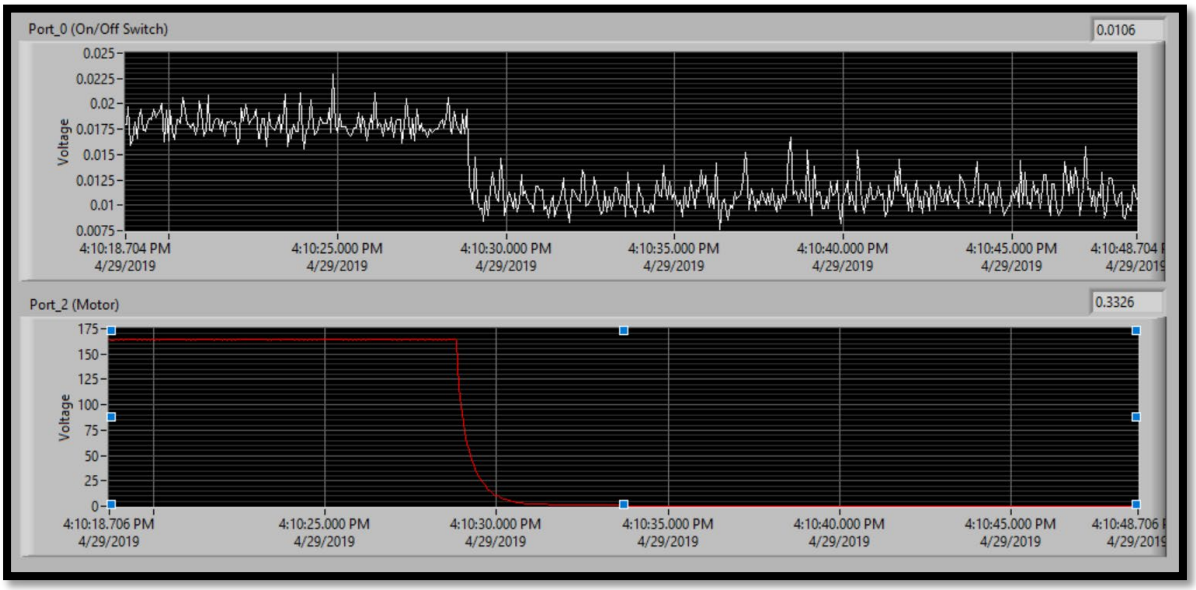


Figure 4-20: Lathe_11 Running, then Motor Lever Switched Off

Figure 4-21 shows the voltage behaviors when the motor is running, then the brake is pressed.

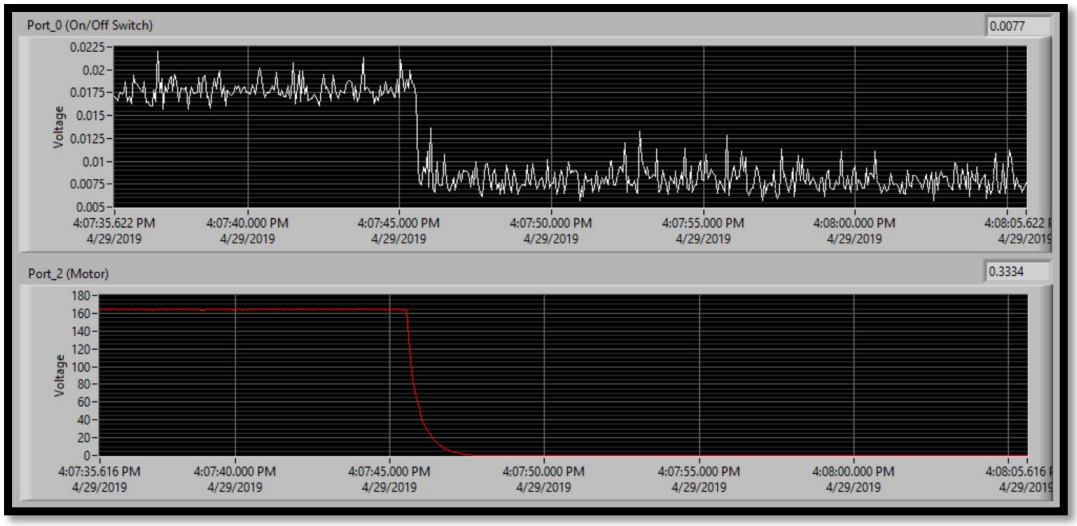


Figure 4-21: Lathe_11 Running, then Brake is Pressed

From this example, it is suggested that it may be unreasonable to try and monitor additional features such as the brake state, light status, etc. Additional ports and/or sensors would be needed to obtain those pieces of information with fidelity.

Another limitation on the lathe real-time monitoring had to do with RPM, a piece of information that the shop supervisor had requested (Table 1-1). Below are figures showing the voltage behaviors being measured as they relate to different RPM rates.

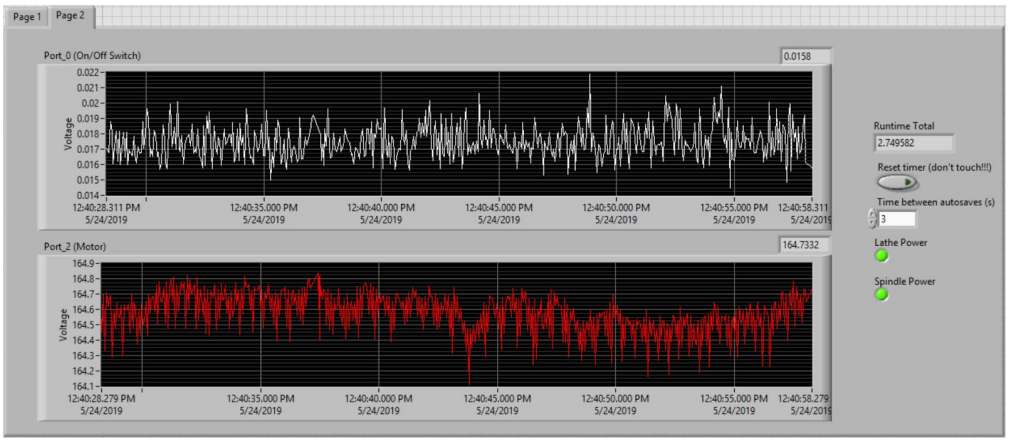


Figure 4-22: Lathe_11 Voltage Conditions at 35 RPM

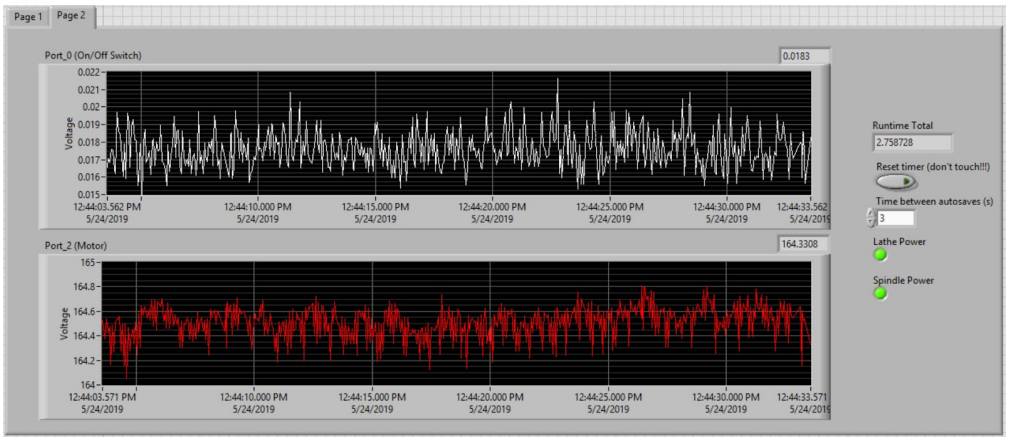


Figure 4-23: Lathe_11 Voltage Conditions at 490 RPM

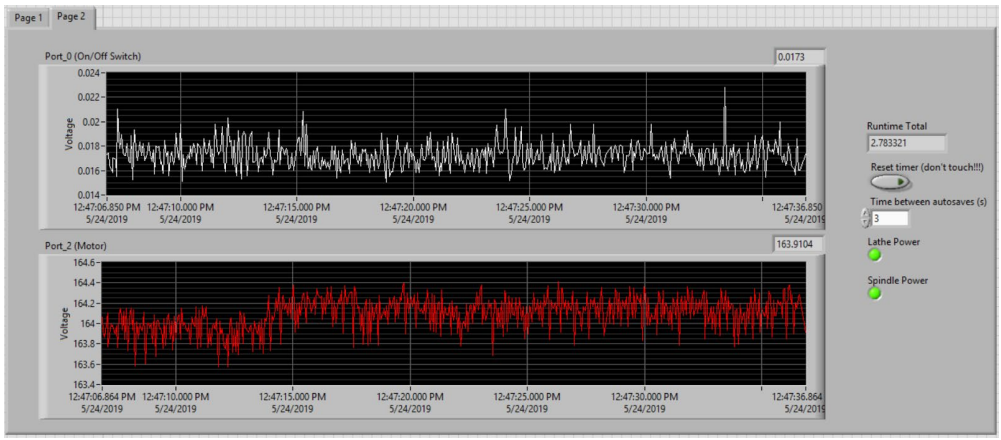


Figure 4-24: Lathe_11 Voltage Conditions at 2000 RPM

From these graphs, it is able to be deduced that there is no substantial change in the voltage values under different RPM conditions. After consideration, this makes sense because electrically everything is remaining the same. The factor that is changing is the gears that are being used to rotate the spindle. RPM would not be able to be measured or derived electrically from this setup.

This then calls for other solutions to be utilized in order to obtain RPM on the lathe if it is going to be incorporated into the smart solution. In Table 1-1, facet 2, it was requested that RPM

be monitored. For this to happen, sensors and other types of hardware would need to be used, but in this research, the intent was to establish real-time monitoring based off of what the machine already has. It is for this reason that RPM will not be pursued to completion in this research, but will provide a good opportunity for future research and experimentation in connecting additional sensors into an already-implemented smart solution.

4.2.3 Summary

Can the information being received from the lathe be reliably manipulated in order to display real-time monitoring?

The information being received from the lathe can be reliably manipulated in order to display real-time monitoring. Statuses and KPIs are able to be measured. However, the level of real-time monitoring in the lathe is limited by the limited amount of raw data being sent to the manufacturing apps. Based off of test results on the most common lathe state conditions, it was determined that features not tied to the machine and motor electrical channels (brake status, RPM, etc.) would not be able to be reliably monitored. Real-time monitoring was able to be established, but facet 2 in Table 1-1 regarding RPM will not be implemented at this time.

4.2.4 Mills

Table 4-5 provides a summary of the information that was able to be obtained in each of the four mills of the smart factory. Significantly more pieces of data were able to be monitored based off of information already being tracked and communicated from the drive to the communication card. Reverse control of the machine was not attempted because of liability reasons.

Table 4-5: Mills Real-Time Monitoring Summary and Analysis

Name	Description	Source Origin	Logic Written	Challenges Encountered	Next Step
Status_sve_trp	Status of machine at time of last trip	FE01 on drive	Read-only tag	Find it in communication manual	n/a
RPMmill1conversion	Derived tag from voltage value	KEPServer	n/a	Calculating the constant	n/a
Output_freq	The frequency the drive is set to	FE00 on drive	Read-only tag	Find it in communication manual	n/a
Outpt_voltage	Incoming voltage of drive from AC reading	FE05 on drive	Read-only tag	Find it in communication manual	n/a
Outpt_curr_disp	Current drawn from motor to meet demand	FE03 on drive	Read-only tag	Find it in communication manual	n/a
Mill1MotorState	Derived tag from KEP.	KEPServer	n/a	n/a	n/a
Mill1MachineState	Derived tag from KEP.	KEPServer	n/a	n/a	n/a
Curr_outpt_freq	The current output freq.	FD00 on drive	Read-only tag	Find it in communication manual	n/a
Curr_freq_comm	Current freq command of the machine	FE02 on drive	Read-only tag	Find it in communication manual	n/a
Cum_run_time	Total spindle run time of machine since its beginning	FE14 on drive	Read-only tag	Find it in communication manual	n/a
Bus_voltage	Incoming voltage from DC reading	FE04 on drive	Read-only tag	Find it in communication manual	n/a
_EStopPressed	System tag automatically generated in KEPServerEX	KEPServer	n/a	n/a (automatically generated)	Find a way to distinguish between e-stop and unavailable
Running	Machine status	Manufacturing apps	(in apps)	n/a	n/a
Unplanned Downtime	Machine status	Manufacturing apps	(in apps)	n/a	n/a
Warning	Machine status	Manufacturing apps	(in apps)	n/a	n/a
Planned Downtime	Machine status	Manufacturing apps	(in apps)	n/a	n/a
Availability	Availability of machine	Manufacturing apps	Run Time / Planned Production Time	Inaccurate default formula	Change default formula
Quality	Quality of parts coming off of line	Manufacturing apps	Good Count / Total Count since beginning of the planned operation time	Education setting (hard code solution)	Set up application for using this
Performance	Ratio of actual run rate to ideal run rate	Manufacturing apps	(Total Count / Run Time) / Ideal Run Rate	Education setting (hard code solution)	Set up application for using this
OEE	Overall Equipment Effectiveness	Manufacturing apps	Availability * Performance * Quality	Inaccurate default formula	Change default formula

Real-time monitoring was attempted and achieved. The following sections and figures, along, with the discussion that follows, help to explain what exactly was achieved in regards to real-time monitoring for the mills. Figure 4-29 shows the additional properties that were able to

be pulled from the inverter on the mill as well as other derived tags used to relay useful data such as revolutions per minute (RPM), machine state, and spindle state. The following four figures depict the linear relationship between the current frequency command tag and RPM.

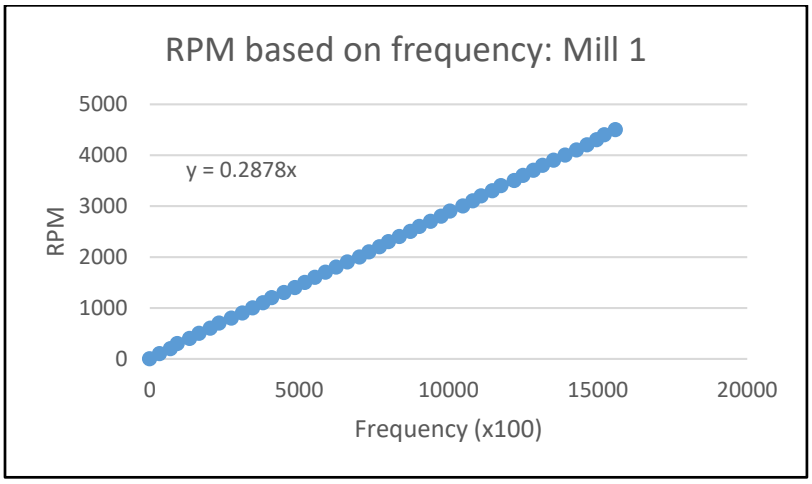


Figure 4-25: Mill 1 RPM Relationship

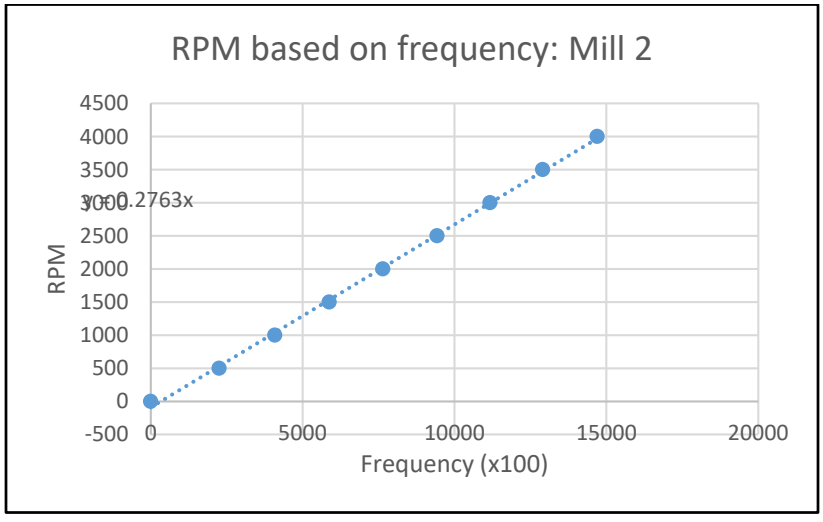


Figure 4-26: Mill 2 RPM Relationship

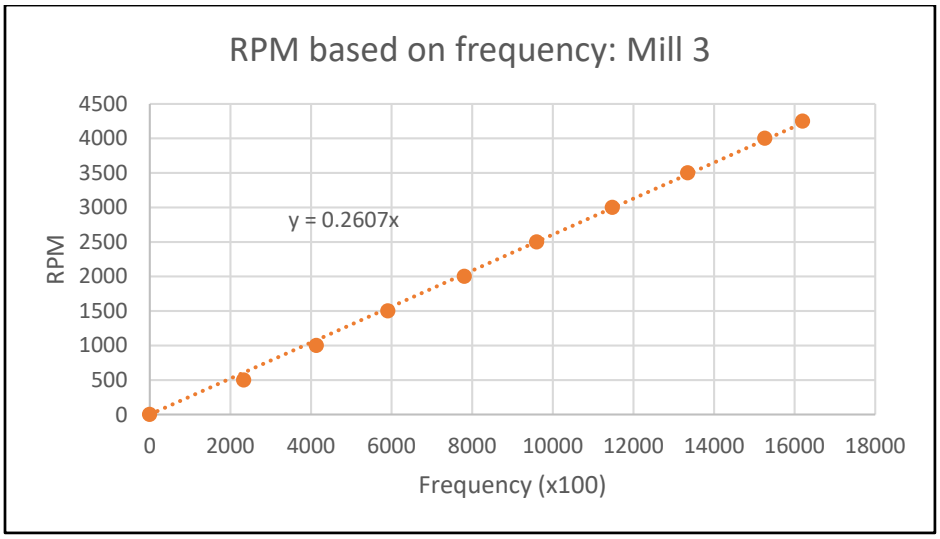


Figure 4-27: Mill 3 RPM Relationship

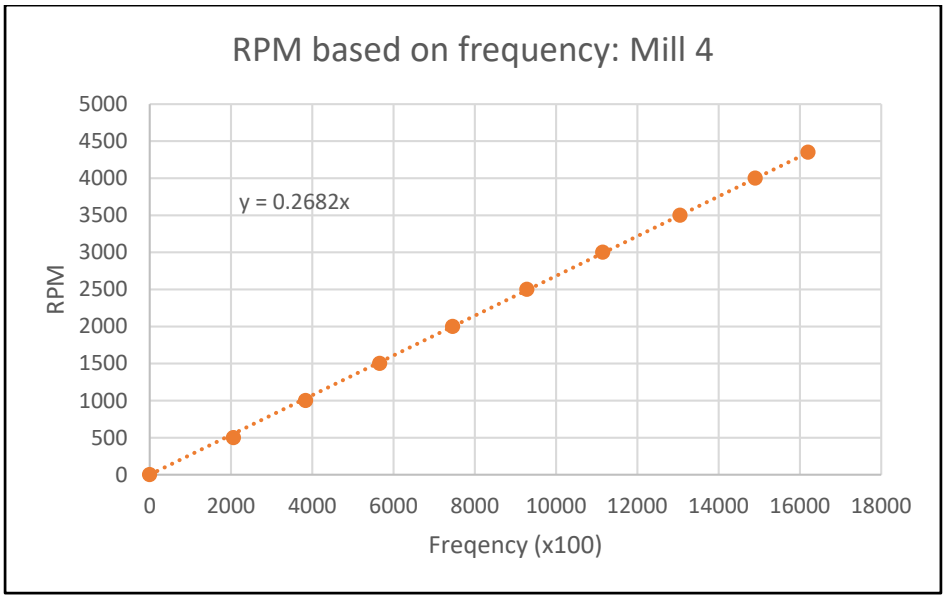


Figure 4-28: Mill 4 RPM Relationship

From tracking the relationship between frequency and RPM on the mill (on the machine itself) a linear constant was able to be found for each of the mills relating the frequency and RPM. Initially, RPM was never being tracked in the tags on the drive. Now that this relationship was established, it did not matter. Now all that needed to be done was to multiply the current

frequency command tag inside of KEPServerEX by the constants found here, set that up to be a derived tag inside of KEPServerEX, and then the RPM would be able to be displayed in the manufacturing apps based off of that derived tag (Figure 4-29). This satisfies facet 2 in Table 1-1 for the shop supervisor's requested pieces of information to have in the smart factory.

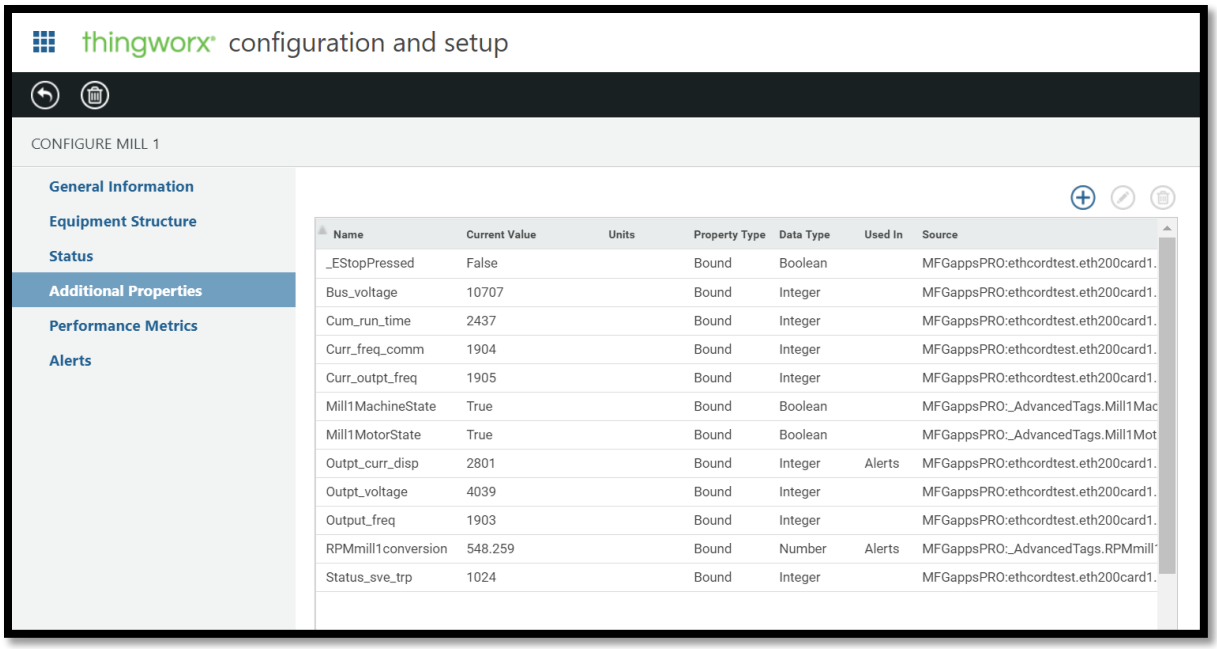


Figure 4-29: Additional Properties for Mill 1 Tied to KEPServerEX Tags

Figure 4-30 shows the statuses of multiple mills. This is accessible under the Asset Advisor tab within the manufacturing apps. Like the lathe, it shows information about the machine, the machine status, how long it has been in that status, if there are any active alerts, and the number of alerts that were triggered that week. This page is configurable using the Customization Guide accessible from the PTC or ThingWorx website, depending on how things are desired to be configured.

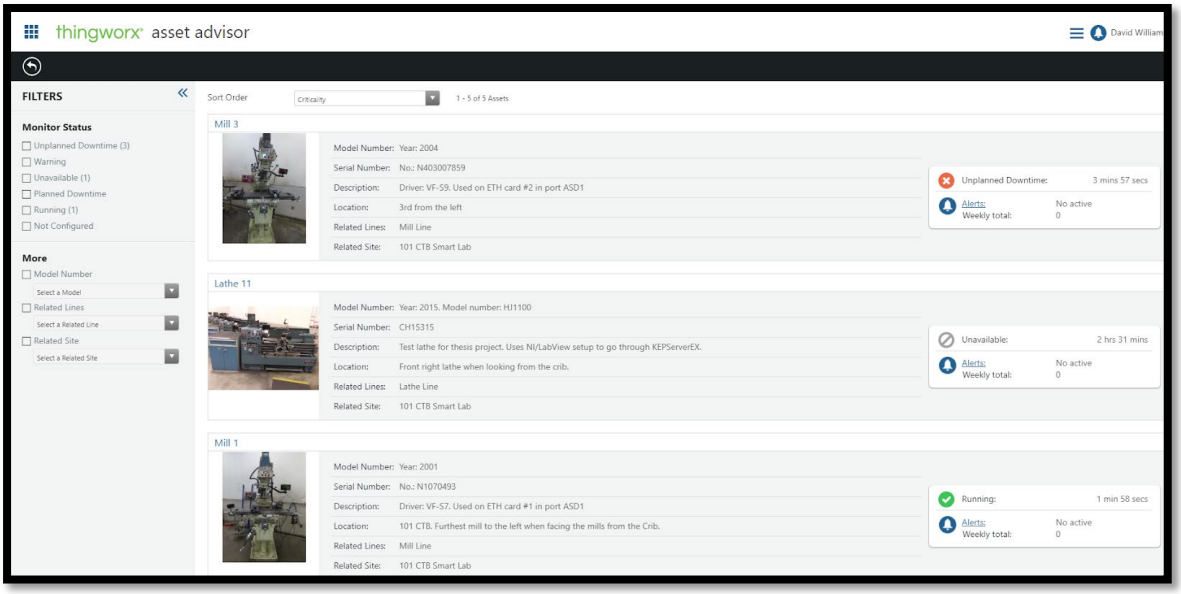


Figure 4-30: Asset Advisor for Mill 1 and Other Assets

Figure 4-31 shows the production key performance indicators (KPIs) of the machines within the factory. It is set up to track overall equipment effectiveness (OEE), availability, quality, and performance. Additionally, it shows the story of the data over time for that KPI calculation period. In this case, the calculation period is 5 minutes. In an industrial setting, it may be more or less.

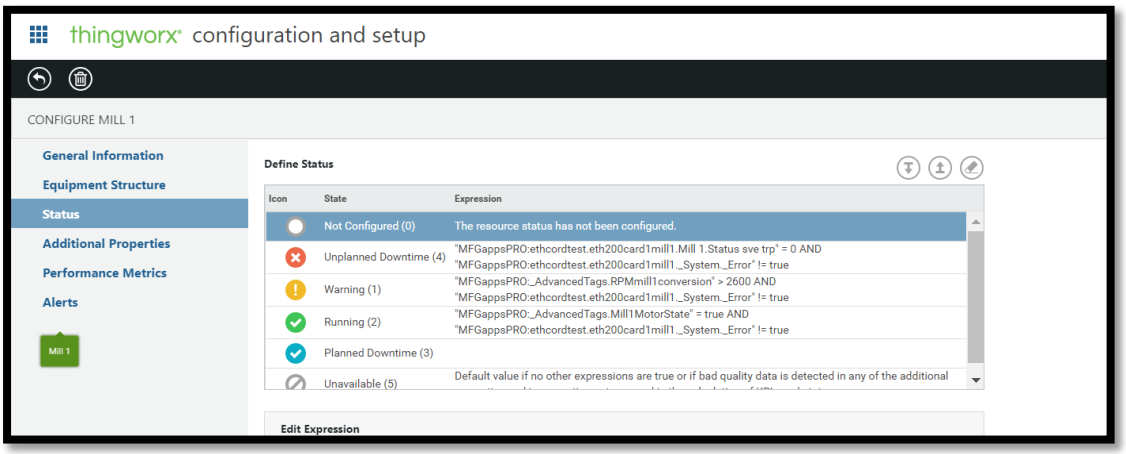


Figure 4-31: Status Definition Tab for Mill 1

Figure 4-32 shows the status definitions for Lathe 11. With the manufacturing apps, the default statuses that can be defined are Running, Unplanned Downtime, Warning, and Planned Downtime. These statuses were based off of the values coming in from KEPServerEX. There was latency in receiving the status updates, but for the educational setting in which this project took place, it was an acceptable update rate (approximately once per minute).

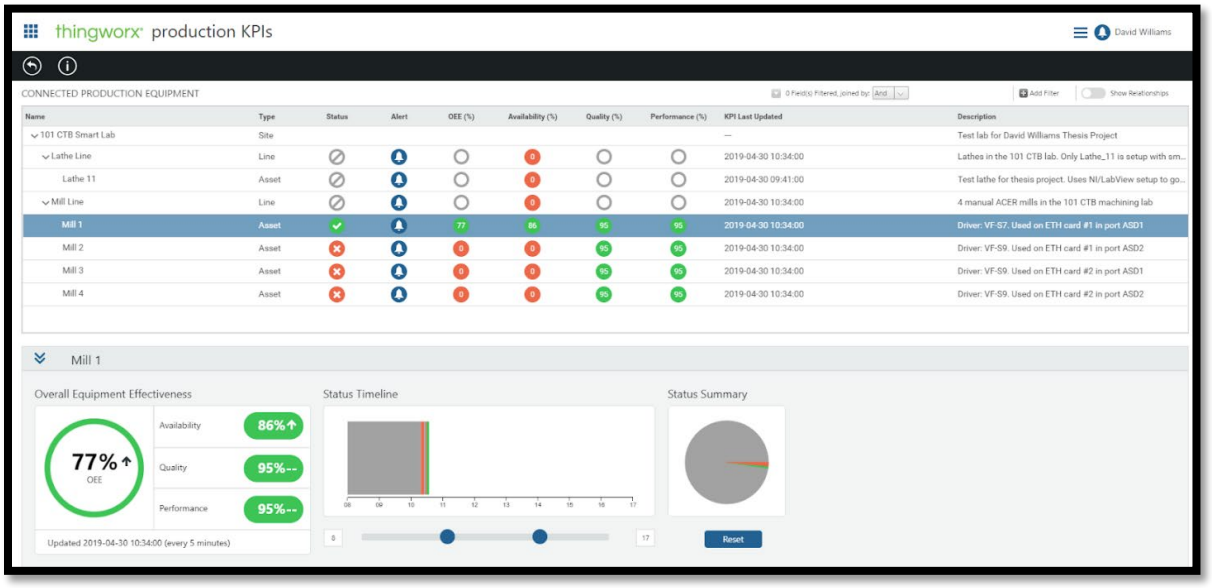


Figure 4-32: Production KPIs View of Mill 1

Figure 4-33 depicts a monitoring of the alerts that have been triggered and are active. This view is found in the Alert Monitoring tab of the manufacturing apps.

Figure 4-33: Alert Monitoring

Figure 4-34 is a screenshot of the historical view within the Alert Monitoring tile of the manufacturing apps.

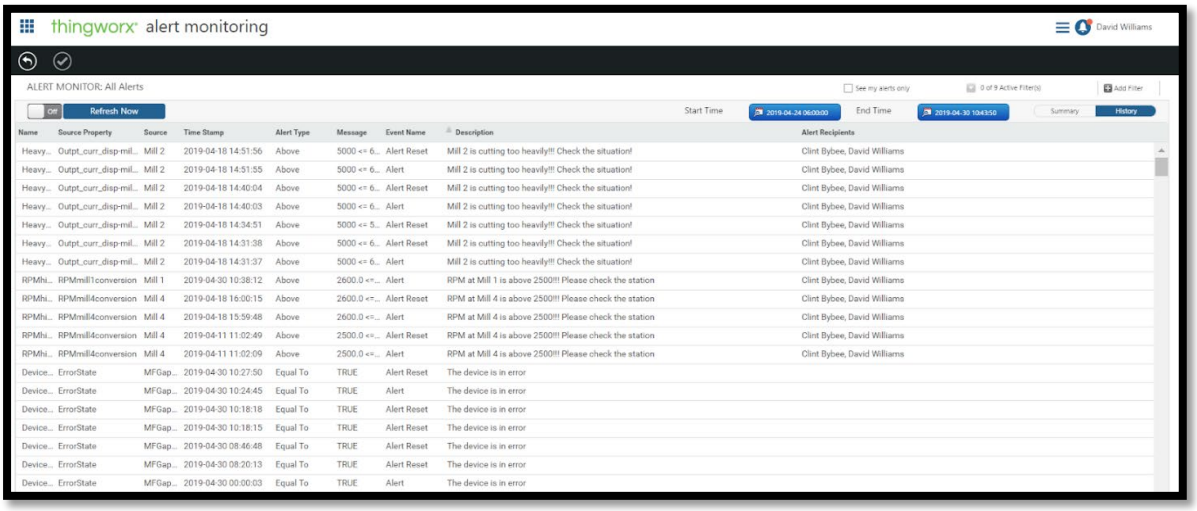


Figure 4-34: Alert Monitoring Historical View

4.2.5 Discussion of Results

4.2.5.1 Advantages

The advantages of the setup of real-time monitoring for the mills in this project are essentially the same as those of the lathe (mentioned above). The manufacturing apps enabled an easy setup and configuration and manipulation of the data into a visual display of data in real-time.

4.2.5.2 Limitations

The limitations of this setup of real-time monitoring for the mills is also essentially the same as the those of the lathe. The monitoring is limited to the scan rate of the machine's

inverter, and the data that can be manipulated and monitored is directly correlated to the amount of raw data that is received in the first place.

4.2.6 Summary

Can the information being received from the mills be reliably manipulated in order to display real-time monitoring?

The information being received from the mills can, in fact, be reliably manipulated in order to display real-time monitoring. The amount of information that was able to be monitored was dramatically more than that of the lathe because the inverter already tracking numerous pieces of information. That information was able to be monitored in real-time and displayed in user-friendly visualization tools of the manufacturing apps. Aspects of these monitored pieces of information include KPI evaluations of all of the mills, alert monitoring, and machine status. Additional tags, such as RPM and machine state tags, were able to be created and monitored based off of linear relationships and other patterns, providing more even more data.

4.3 Issue Identification

The following sections describe in detail the results of implementing issue identification capabilities into the smart system set up of the lathe and the four mills in this project. These results are based off of code written in ThingWorx Composer and alerts configured in the manufacturing apps. All of these are in response to the data that is being brought in from the lathe and mills of the smart set up. The automated notifications that are provided below are common alerts to be found in industry. Other features may be incorporated in other applications, but these were the features desired off of these machines.

4.3.1 Lathe

Table 4-6 outlines the numerous issues and anomalies that were interpreted and responded to by the system for the lathe. These issues were incorporated specifically to address the kinds of issues identified in Table 1-1.

Table 4-6: Lathe Issue Identification Summary and Analysis

Anomaly	Signal Used	Logic of Processing	Action Taken	Challenges Encountered	Next Steps
After-hour operation	MachineOnMotorOn tag	Boolean value	Email, text	Hiccups in REST API version	Text
Lost connection	System_error tag	Boolean value	Email, text	Days between occurrences	Code for resetting program
Heavy cutting/tool crash	Port_2 tag	Defined threshold	Email, text	Multiple emails sent, uncontrollable voltage drops	Code for single email

Figure 4-35 shows an automated email that was sent in response to the lathe being on after hours. This notification was triggered off of a service written in ThingWorx Composer. This service is triggered at a certain specified time each day, and is enabled/disabled at other specified times during the day. The service scans to see if the property values associated with the spindle is on. If that value is true, then the automated alert is sent. The email account from which the email is sent, as well as the subject details is all configurable inside of ThingWorx Composer and is able to be adjusted based off of the different alert type. This scan simply discerns if the Boolean value of the MachineOnMotorOn tag is true or not. Because the value was evaluated to be true, the following email was sent.

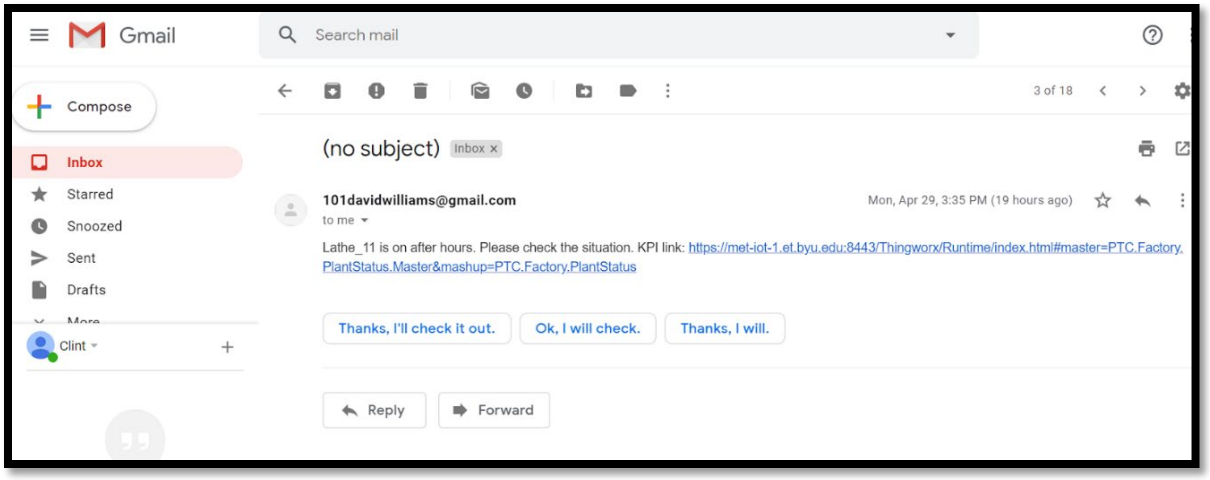


Figure 4-35: After-Hour Email Notification for Lathe_11

Figure 4-36 shows the email notification that is sent in the event that there is an error in the connection (due to the LabVIEW program being ended/aborted). This notification was set up as an alert inside the manufacturing apps. This provides a convenient solution for knowing exactly when a machine is down, and which machine that may be.

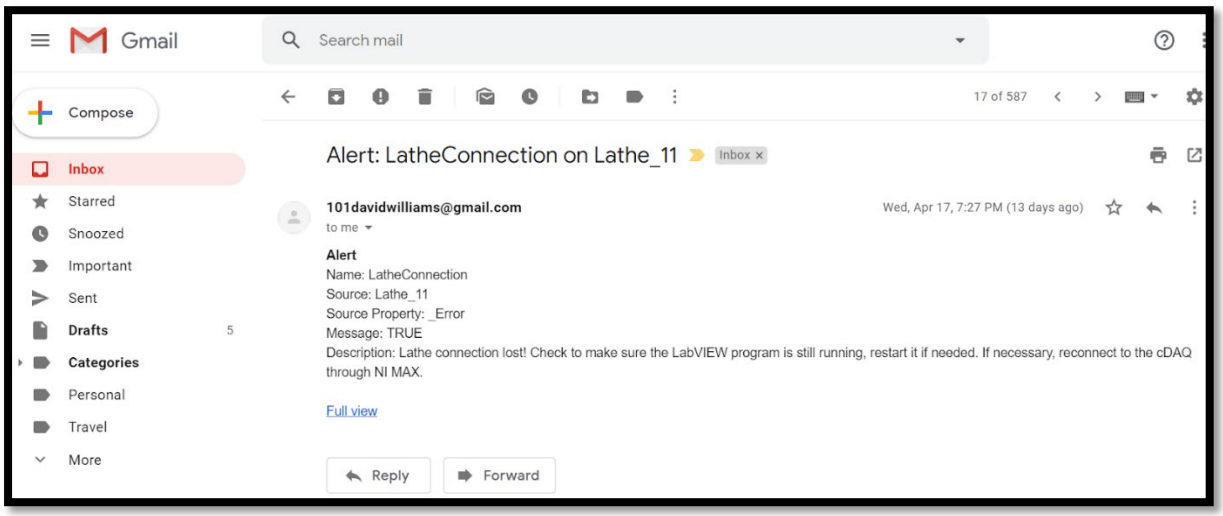


Figure 4-36: Email Notification for Lost Lathe Connection

4.3.2 Discussion of Results

4.3.2.1 Limitations

All of the above anomaly notifications were fairly easy to construct and configure. However, a notification in the event of a crash or heavy cutting on the lathe would not be as reliable. The reason is for this is because of the similarity in behavior in the event of heavy cutting and the voltage behavior when another lathe or machine in the shop is turned on. Figure 4-37 depicts lathe 11 undergoing heavy cutting (simulated crash). It is seen that there is an approximate 3-volt drop.

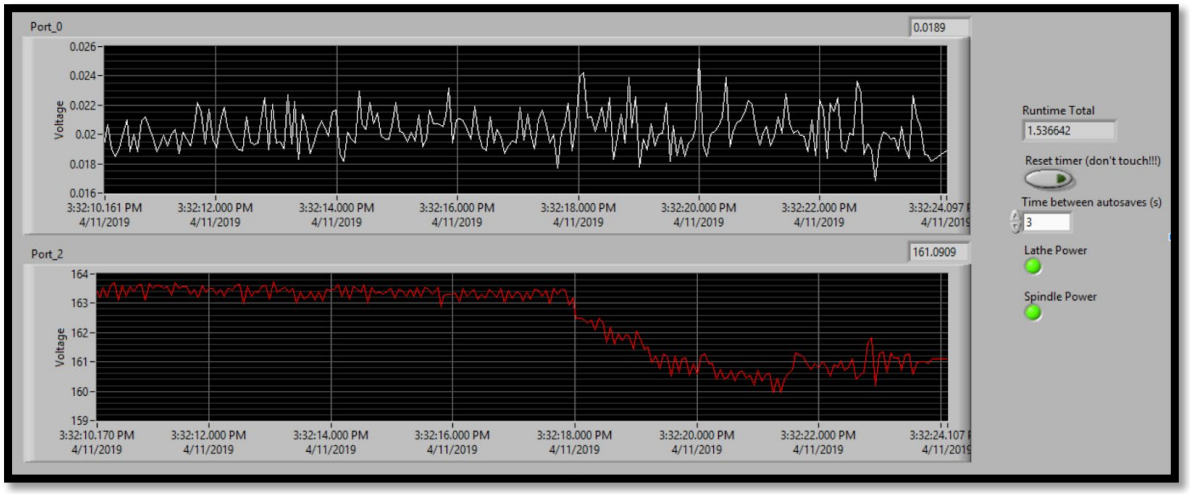


Figure 4-37: Lathe_11 in a Heavy-Cutting State

Figure 4-38 shows the voltage behavior of Lathe 11 running, and then another lathe (Lathe 10) being turned on while it is running. This test was done to help demonstrate some of the probable conditions and circumstances to which the lathe will be subjected. If there are multiple machines, then the voltage change would likely be even greater.

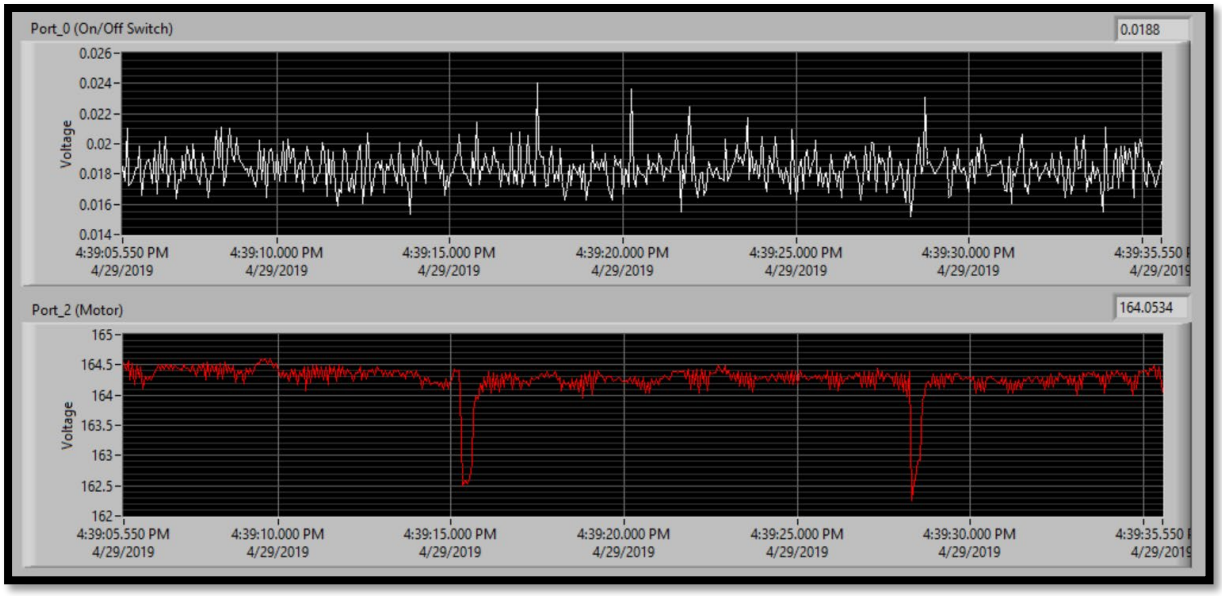


Figure 4-38: Lathe_11 Running, then Lathe 10 Turned On

As seen in the figure above, there is approximately a 2-volt drop. This drop is from only 1 other lathe/machine being turned on while the spindle on Lathe 11 is running. It is safely assumed that if more than 1 machine is in use or is turned on, the voltage drop in this figure would be greater. The behaviors are different (one is much sharper than the other) however, the amount of voltage change is what is being watched. Thresholds are set to determine the state of the machine. If a ± 2.5-volt threshold is placed on the lathe system, the probability of receiving false-positives greatly increases.

4.3.3 Summary

From the data being transferred and monitored, can issue identification be implemented in the lathe to alert shop supervisors in the event of anomalies?

Once the data is coming into the manufacturing apps or ThingWorx Composer, logic was able to be written and alerts were able to be configured to reliably send an automated notification

in the event of several anomalies: after-hour operation, and lost connection. The alert to be sent in the event of heavy cutting or a crash, the voltage behaviors of the raw data under heavy cutting conditions and the behavior when another machine was turned on were too similar to reliably trigger a true-positive of a machine crash. Each of the alerts also answered the alerts desired by the shop supervisor.

4.3.4 Mills

The following sections with their associated figures represent the results of integrating issue identification capabilities into the smart system for the mills. Table 4-7 summarizes the alerts that would automatically notify shop personnel in the event of each respective alert becoming active. These alerts were implemented in response to the feedback provided by the shop supervisor, given in Table 1-1.

Table 4-7: Mills Issue Identification Summary and Analysis

Anomaly	Signal Used	Logic of Processing	Action Taken	Challenges Encountered	Next Steps
After-hour operation	MotorState tag	Boolean value	Email, text	n/a	Text
Lost connection	ErrorState property	Boolean value	Email	Distinguishing between lost connection and e-stop	Creating e-stop status
Heavy cutting/tool crash	Outpt_curr_disp property	Defined threshold	Email, text	Different part materials changes the threshold	Standardizing the desired threshold
High RPM	RPMmillconversion tag	Defined threshold	Email, text	Linking Twilio trial account to notification	Obtaining lasting twilio account for system

Figure 4-39 shows the active alert icon within the Asset Advisor of the manufacturing apps.

Figure 4-39: Alert Monitoring for Mill 1

Figure 4-40 shows the email notification that was sent in the event of a high RPM on Mill 1. The alert to send this automated notification was built inside of the manufacturing apps. It is based off of the RPMmill1conversion property being received from KEPServerEX and is configurable for a shop supervisor.

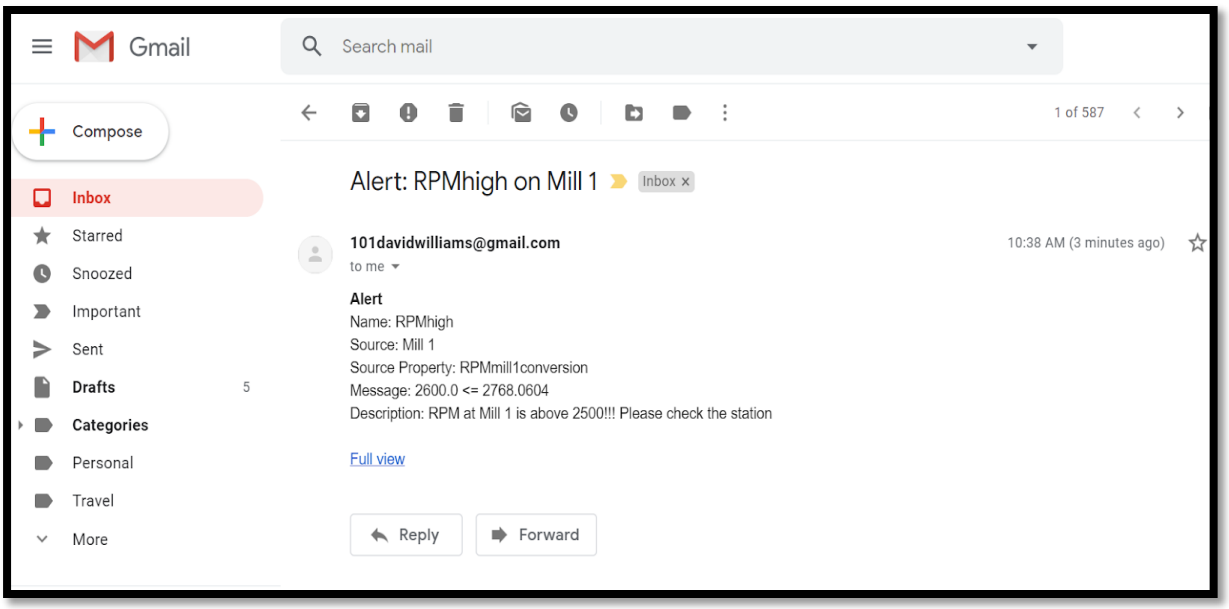


Figure 4-40: Email Notification for High RPM on Mill 1

Figure 4-41 is a screenshot of the email notification that was sent in the event of heavy cutting on Mill 2. This alert/notification was configured in the manufacturing apps and is based off of the derived tag coming from KEPServerEX.

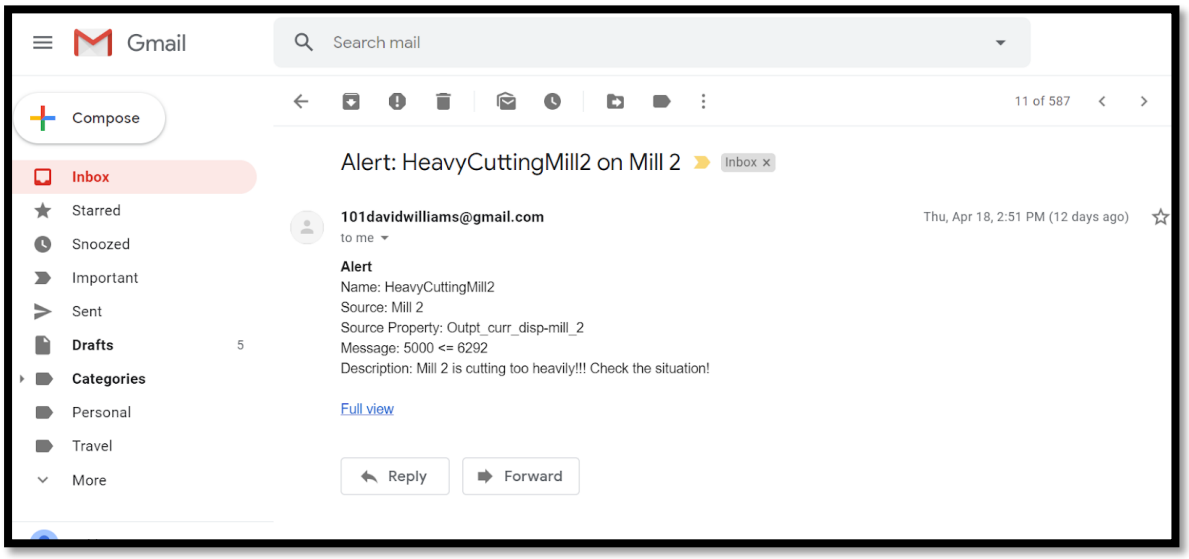


Figure 4-41: Email Notification for Heavy Cutting On Mill 1

Figure 4-42 shows the email that was sent in the event of Mill 1 being on after hours. This notification was set up and configured within ThingWorx Composer. Written in the exact same service used to scan if the lathe was on, this machine was included in that service and would trigger a notification to be sent to the shop supervisor in the event that it is on outside of normal operating hours. Corresponding alerts and emails were created and tested for the other mills in the smart factory as well. This email automatically comes with a link to the manufacturing apps, allowing access to that email recipient to look into the apps and see in real-time the status of the machine.

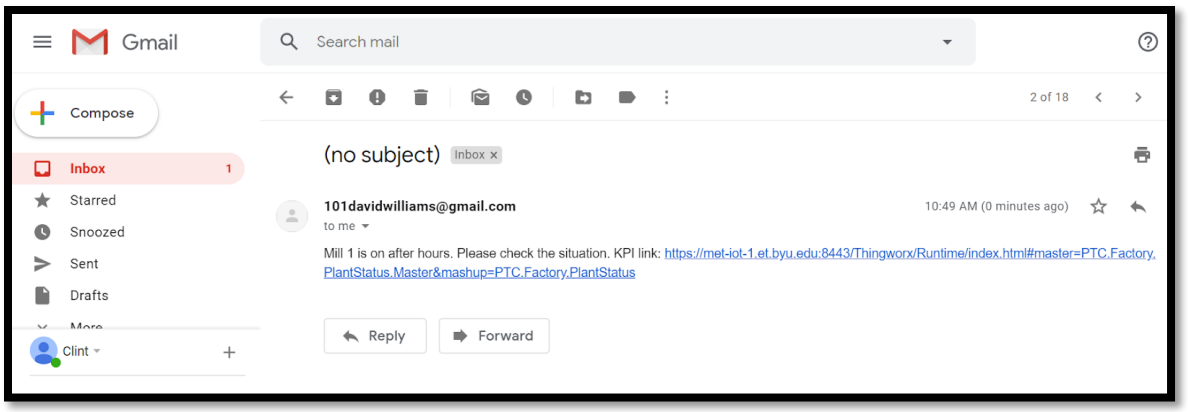


Figure 4-42: Email Notification for After-Hour Operation on Mill 1

4.3.5 Discussion of Results

From the efforts that were made concerning issue identification, there were two main methods of sending alerts in the event of a machine issues or anomalies: custom-made email notifications in ThingWorx Composer, and automated emails coming from the ThingWorx manufacturing apps. Both methods were used in this project.

For the custom-made emails from Composer, the alert for after-hour machine operation was done. This was done in Composer because the apps did not have a service that scanned the machines for after-hour operation. This helps to demonstrate the flexibility of the system and what is able to be accomplished. The service that was constructed to scan for after-hour operation was designed to only scan once per specified unit of time.

The manufacturing apps handled all of the other alert notifications. When an alert was triggered, the corresponding email with an alert description and link to the apps was sent to the appropriate user. This proved to be very reliable and only sent one notification when that alert state was reached. When the alert state was cleared, notifications would not be sent until another alert was triggered.

4.3.6 Summary

From the data being transferred and monitored, can issue identification be implemented in the mills to alert shop supervisors in the event of anomalies?

Once the data is coming into the manufacturing apps or ThingWorx Composer, logic was able to be written and alerts were able to be configured to reliably send an automated notification in the event of several anomalies: after-hour operation, and a high RPM. Additionally, a reliable notification was able to be created for the mills that would notify the shop supervisor of heavy cutting, or a crash. Other notifications may be constructed in the future, but these were the features to test for issue identification capabilities, and were constructed in response to the request of the shop supervisor.

4.4 Shop Supervisor Requested Items Summary

Table 4-8 summarizes the results of satisfying the requests of the shop supervisor for the lab containing the smart factory setup.

Table 4-8: Results on Shop Supervisor Requests

Desired data			
Data Requested	**Implemented on Lathe (yes/no)**	**Implemented on Mills (yes/no)**	**Reasons for not being implemented/recommendations**
1. Machine/spindle-spinning or stopped	Yes	Yes	n/a
2. Spindle RPM	No	Yes	This would require additional sensors (outside of scope)/ look into sensors that would be able to connect into National Instruments or directly into KEPServerEX
3. Spindle load	No	No	This would require additional sensors (outside of scope)/ look into force sensors compatible with KEPServerEX
4. Crash notification	Yes	Yes	n/a
5. How many hours the spindle is in operation	Yes	Yes	n/a

As shown in the table above, most of the criteria were met in the completion of this project. The only criteria that were not met were spindle RPM on the lathe, and spindle load on both the lath and mill. All other requirements were met, including all of the desired alert notifications, machine status, and spindle status aspects. The reason that the other two criteria were not completed or pursued further was because the solutions to these would start to venture outside of the scope of this research. This research intended to answer if the integration of these machines into a larger smart system would be feasible, and evidence supports this. RPM and spindle load fall more appropriately into the "features" category in which they would be nice to have and add on, but are not necessary in order to answer or demonstrate proof of smart factory integration. They do, however, provide important pieces of information that should be acquired in future research and setups and will more than likely be implemented at a later date.

5 CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

5.1.1 Unified Connectivity

Is it possible to establish unified connectivity between manual lathes/mills and a computer?

5.1.1.1 Lathe

From the setup and results of the lathe portion of this project, it is possible to establish unified connectivity between a manual lathe and a computer. The lathe connectivity was accomplished by measuring and acquiring data using National Instruments devices and having its control program, LabVIEW, send that data to a computer to a commonly used piece of software in the manufacturing industry called kepware (KEPServerEX) via an OPC UA connection. 24/7 connectivity was set up with the ability to send its data to a visualization tool on an IIoT platform called the ThingWorx Manufacturing Apps. Although setup was costly, the principles and steps used prove valuable for implementation of other types of NI products across the manufacturing industry everywhere. From the evidence provided, it can be reasonable assumed that unified connectivity is indeed able to be established and allow a lathe to become fully integrated as part of a larger smart system.

5.1.1.2 Mills

From the setup and procedures shown, it was discovered that it is indeed possible to establish unified connectivity between manual mills and a computer. With this setup, a quick and easy solution to establish unified connectivity was constructed to a device that was not initially meant to be tracked and monitored using a relatively inexpensive communication card to convert Toshiba protocol into a more common Modbus TCP/IP protocol.

5.1.2 Real-Time Monitoring

Can the information being received from these machines be reliably manipulated in order to display real-time monitoring?

5.1.2.1 Lathe

The information being received from the lathe can be reliably manipulated in order to display real-time monitoring. That information is able to be displayed in a variety of ways to determine if the machine is on, running, in planned downtime, in unplanned downtime, offline, etc. The KPIs are able to be measured off of the lathe. However, the level of real-time monitoring in the lathe is limited by the limited amount of raw data being sent to the manufacturing apps. Based off of test results on the most common lathe state conditions, it was determined that features not tied to the machine and motor electrical channels (brake status, RPM, etc.) would not be able to be reliably monitored. Real-time monitoring was able to be established, but facet 2 in Table 1-1 regarding RPM will not be implemented at this time.

5.1.2.2 Mills

The information being received from the mills can, in fact, be reliably manipulated in order to display real-time monitoring. The amount of information that was able to be monitored was dramatically more than that of the lathe because the inverter already tracking numerous pieces of information. That information was able to be monitored in real-time and displayed in user-friendly visualization tools of the manufacturing apps. Aspects of these monitored pieces of information include KPI evaluations of all of the mills, alert monitoring, and machine status. Additional tags, such as RPM and machine state tags, were able to be created and monitored based off of linear relationships and other patterns, providing more even more data.

5.1.3 Issue Identification

From the data being transferred and monitored, can issue identification be implemented to alert shop supervisors in the event of anomalies?

5.1.3.1 Lathe

Once the data is coming into the manufacturing apps or ThingWorx Composer, logic was able to be written and alerts were able to be configured to reliably send an automated notification in the event of several anomalies: after-hour operation, and lost connection. The alert to be sent in the event of heavy cutting or a crash, the voltage behaviors of the raw data under heavy cutting conditions and the behavior when another machine was turned on were too similar to reliably trigger a true-positive of a machine crash. Each of the alerts also answered the alerts desired by the shop supervisor.

5.1.3.2 Mills

Once the data is coming into the manufacturing apps or ThingWorx Composer, logic was able to be written and alerts were able to be configured to reliably send an automated notification in the event of several anomalies: after-hour operation, and a high RPM. Additionally, a reliable notification was able to be created for the mills that would notify the shop supervisor of heavy cutting, or a crash. Other notifications may be constructed in the future, but these were the features to test for issue identification capabilities, and were constructed in response to the request of the shop supervisor.

5.2 Recommendations for Future Research

For the lathe connection, it would be useful to try and find more ways to apply additional sensors so that RPM, spindle load, light status, coolant status, spindle forward/spindle reverse, etc. could be tracked and monitored. It would also be useful to see if there is a way to tap into the monitor that displays the position of the tool and have that be monitored in the manufacturing apps as well.

Additional pieces of data will allow for more monitoring and a more comprehensive view of the state of the machine. It will also allow for more anomalies to be tracked.

For the mills, it would be beneficial to the shop supervisors or TAs to be able to monitor the spindle direction, the table position, and whether or not the spindle is in high gear or low gear. They both have ranges in which they operate safely, and being able to tell which gear the machine is in would help provide accurate alerts and notifications with a more detailed explanation of what is occurring.

From the aspect of the manufacturing apps, it would be useful to customize the KPIs to have visuals of the information that is desired by the shop supervisors. The manufacturing apps are naturally built for the mass manufacturing setting. Finding a way to customize the apps for any environment would be beneficial for industry as well in the event that customization is needed

REFERENCES

Bates, Jeff. 2018. "Merging Legacy Equipment with the Industrial Internet of Things: Three Approaches for Integrated Data."

Biron, Joe, Don Busiek, and Jon Lang. 2018. "The State of the Industrial Internet of Things: A Spotlight on Industrial Innovation."

Ivester, Robert W., and Jarred C. Heigel. 2000. *Smart Machining Systems : Robust Optimization and Adaptive Control Optimization for Turning Operations*. vol. Book, Whole. Dearborn: Society of Manufacturing Engineers. https://ebookcentral.proquest.com/lib/[SITE_ID]/detail.action?docID=3337873.

Kepware. 2019. "Thingworx Native Interface." https://www.kepware.com/en-us/products/kepserverex/features/thingworx-native-interface/.

Kumar, Manoj, Rahul Vaishya, and Parag. 2018. "Real-Time Monitoring System to Lean Manufacturing." *Procedia Manufacturing* 20 (2018/01/01/): 135-140. https://dx.doi.org/https://doi.org/10.1016/j.promfg.2018.02.019.

McAvoy, Jack. 2019. PTC. 2019. Form of Item. https://www.businesswire.com/news/home/20181101005795/en/PTC-Ranked-Number-IoT-Platforms-Report-Rapid.

McKewen, Ellen. 2015. "What Is Smart Manufacturing? (Part 1a)." Form of Item. https://www.cmtc.com/blog/what-is-smart-manufacturing-part-1a-of-6.

Muhuri, Pranab K., Amit K. Shukla, and Ajith Abraham. 2019. "Industry 4.0: A Bibliometric Analysis and Detailed Overview." *Engineering Applications of Artificial Intelligence* 78: 218-235. https://dx.doi.org/10.1016/j.engappai.2018.11.007.

Petrova-Antonova, Dessislava, Georgi Andreev, and Sylvia Ilieva. *Unified Connectivity of Iot Devices through Abstraction of Application Protocols*: ACM.

Porter, Michael E., and James E. Heppelmann. 2014. "How Smart, Connected Products Are Transforming Competition." *Harvard business review*. http://www.econis.eu/PPNSET?PPN=804389543.

---. 2015. "How Smart, Connected Products Are Transforming Companies." *Harvard business review*. http://www.econis.eu/PPNSET?PPN=837835453.

PTC. 2017. "13 Considerations for Your Warehouse Management System."

---. 2018. *3d Systems Accelerates Iot Initiatives with Thingworx Iiot Platform*: PTC.

Qi, Qinglin, and Fei Tao. 2018. "Digital Twin and Big Data Towards Smart Manufacturing and Industry 4.0: 360 Degree Comparison." *IEEE Access* 6: 3585-3593. https://dx.doi.org/10.1109/ACCESS.2018.2793265.

Roblek, Vasja, Maja Mesko, and Alojz Krapez. 2016. *A Complex View of Industry 4.0 April-June 2016*.

Statista. 2018. "Internet of Things (Iot) Connected Devices Installed Base Worldwide from 2015 to 2025 (in Billions)." https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/.

Techopedia. 2019. "Smart Device." 2019. Form of Item. https://www.techopedia.com/definition/31463/smart-device.

Thingworx. 2018. "Ptc; Colfax Selects Thingworx Platform on Microsoft Azure to Accelerate Iot Initiatives across Its Businesses." *Journal of Engineering* (Atlanta), NewsRx. https://search.proquest.com/docview/1999427238.

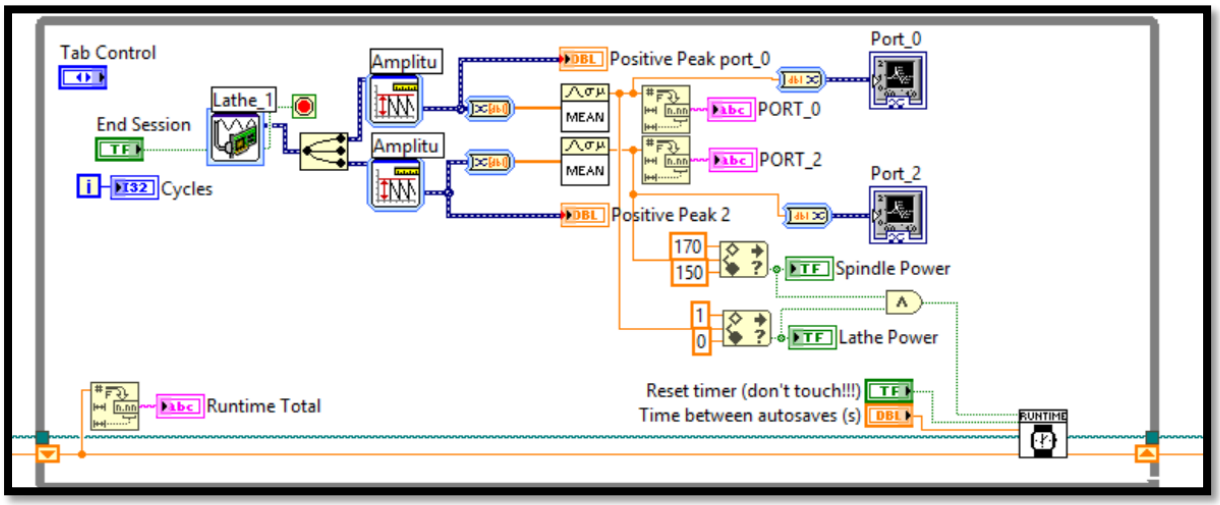
Wang, Jinjiang, Yulin Ma, Laibin Zhang, Robert X. Gao, and Dazhong Wu. 2018. "Deep Learning for Smart Manufacturing: Methods and Applications." *Journal of Manufacturing Systems* 48: 144-156. Accessed Jul. https://dx.doi.org/10.1016/j.jmsy.2018.01.003.

Wrenn, Kevin, and Brian Thompson. 2018. "Designing Smart Connected Products."

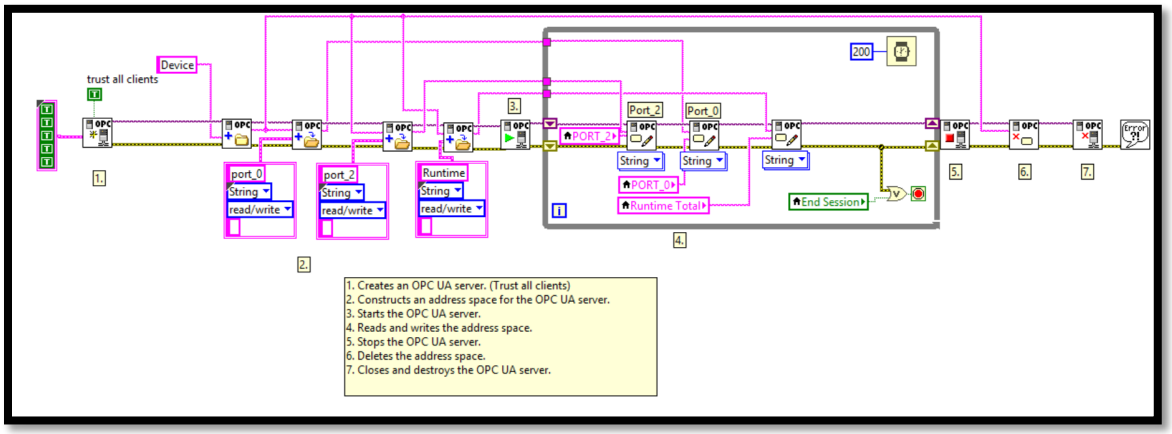
Zhong, Ray, Xun Xu, Eberhard Klotz, and Stephen Newman. 2017. "Intelligent Manufacturing in the Context of Industry 4.0: A Review."

APPENDIX A: LABVIEW CODE

LabVIEW code written to acquire the voltage values from port_0, port_2, and the runtime total



LabVIEW code used to establish OPC UA connection with KEPServerEX



LabVIEW code written and used in initial connection to ThingWorx using Rest API