A SECURE MTCONNECT COMPATIBLE IOT PLATFORM FOR MACHINE MONITORING THROUGH INTEGRATION OF FOG COMPUTING, CLOUD COMPUTING, AND COMMUNICATION PROTOCOLS

A Dissertation
Presented to
The Academic Faculty

by

Mahmoud Parto

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in Mechanical Engineering in the
George W. Woodruff School of Mechanical Engineering

Georgia Institute of Technology
December of 2017

A SECURE MTCONNECT COMPATIBLE IOT PLATFORM FOR MACHINE MONITORING THROUGH INTEGRATION OF FOG COMPUTING, CLOUD COMPUTING, AND COMMUNICATION PROTOCOLS

Approved by:

Dr. Thomas R. Kurfess, Advisor
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Christopher J. Saldana
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Shreyes N. Melkote
School of Mechanical Engineering
Georgia Institute of Technology

Date Approved: November 17th 2017

ACKNOWLEDGEMENTS

I would like to especially thank my mother and father, without whose guidance and support I would not be here.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

SM Smart Manufacturing

IoT Internet of Things

CNC Computer Numerical Control

PLC Programmable Logic Controller

API Application Programming Interface

MCU Microcontroller Unit

IAN Internet Area Network

LAN Local Area Network

AWS Amazon Web Services

UART Universal Asynchronous Receiver-Transmitter

RS Recommended Standard

TTL Transistor–Transistor Logic

TCP Transmission Control Protocol

UDP User Diagram Protocol

XML Extensible Markup Language

JSON JavaScript Object Notation

OEE Overal Equipment Effectiveness

REST Representational State Transfer

HTTP Hypertext Transfer Protocol

DDS Data Distribution Service

AMQP Advanced Message Queuing Protocol

CoAP Constrained Application Protocol

MQTT Message Queue Telemetry Transport

SUMMARY

Industry 4.0 aims at utilizing advancing computational frameworks such as Human-Computer Interaction and Machine Learning in traditional areas of manufacturing and production. Smart Manufacturing (SM) involves creating intelligent manufacturing systems that use the concepts of Industry 4.0 throughout a product development lifecycle to react to design changes with minimal negative impacts on time and cost of manufacturing. SM often relates to Internet of Things (IoT) and Cyber-Physical Systems (CPS) in the way that hardware and software facilitate the communication of actions in different parts of a manufacturing system. A common method of communication among different parts of a SM system is the Internet. Sensors, platforms, and services that communicate in SM need to securely connect to the Internet and communicate with one another in a standard language. An increasingly popular language for communication in IoT in general and more specifically for hardware in manufacturing is MTConnect. The goal of this work is to demonstrate an approach for development of a platform that collects high frequency data from MTConnect and non-MTConnect platforms, enables machines and platforms to directly communicate with one another, processes the collected data, and securely communicates with Internet and cloud without the need of a static IP address. More specifically, the proposed platform consists of two separate sections, a Local Area Network (LAN) and an Internet Area Network (IAN). The LAN communicates with machines via MTConnect, performs fog computing, and transfers the data to the IAN. The IAN section receives the data from LAN while acquiring high frequency data from sensors. This platform then communicates with Internet-connected devices and web APIs for

different tasks such as inserting data to a database, providing data for web apps or smartphone apps, sending alerting messages, and communicating with cloud services such as Google Clouds, Amazon Web Services, If-This-Then-That (IFTTT), and Particle Clouds.

1. INTRODUCTION

In recent decades, digital manufacturing and integration of software to manufacturing systems have been playing essential roles in automation and smart manufacturing (SM). Industry 4.0 that integrates Cyber-Physical System (CPS), Internet of Things (IoT), and Cloud Computing (CC), is an advanced concept that promises a new industrial revolution in manufacturing by enabling full communications between all involved recourses. This revolution follows the three past revolutions in the manufacturing industry that are known as mechanization or Industry 1.0, mass production or Industry 2.0, and automation or Industry 3.0 [1]. The major goal of industry 4.0 lies in the application of the Internet as the means of communication of hardware and web applications that interact with them for data collection and processing [1]. This real-time communication allows the systems to interact with other parts of the system, and make decisions to improve the overall process. For instance, a possible failure detected by an embedded system in the machine can inform the other parts of the equipment to behave differently to prevent the failure or reduce the possible damages. Since the range of communication is dependent on the network size, providing Internet to all parts of the system not only increases this accessibility significantly, but also it opens up endless possibilities such as cloud computing and interacting with web apps, which may not be achievable in traditional methods of local communication. The goal of integration of CC and IoT, as the core supporting technologies in Industry 4.0, is to convert shop-floor resources such as sensors, machines, work processes, and materials to smart objects, as well as empowering workers with more reliable and flexible decision-making tools. These objects, as a result, will be able to act

and react with one another in a cloud-based intelligent environment [2-4]. This model is known as a Digital Twin which provides a digital representation of the shop floor recourses on the cloud enabling analytics of the data, simulation of the health conditions of machines, and optimization of the production processes [5]. Achieving the goals of Industry 4.0 requires addressing many problems and challenges. Despite significant achievements in the existing solutions, they still have many limitations. One of the primary issues is related to the wide variety of communication protocols for the systems, which causes difficulties in easy and proper interactions. Devices and platforms with different communication protocols cannot directly communicate with one another, for instance. Additionally, the security of these communications, especially for connecting sensitive and highly expensive equipment to the Internet, faces significant challenges as cyber-attacks permeating every segment of public and private entities connected to the Internet. This thesis proposes an approach for development of a secured and cost effective IoT platform for machine monitoring through integration of fog computing, cloud computing, and communication protocols, which facilitates the collection, exchange, and analysis of the data from all systems in a machine shop. In this thesis, the remaining sections are presented as follows: in the Background, an overview of the sensors and platforms, common communication protocols, and literature of this work are reviewed. Next, in the Proposed Framework section, an approach is presented that addresses the limitations of the other related works. In the Implementation and Discussion section, an example platform based on the proposed framework is shown and the outcomes are discussed. Finally, in the Contribution and Conclusion the significant of this study, cons and pros of the proposed method, and recommendations for future works in this area are provided.

2. BACKGROUND

Design and development of infrastructure to achieve the goals of Industry 4.0 require consideration of many areas. These areas include methods of unified data collection, communication, connectivity, and accessibility, as well as cloud storage, data analytics, and data visualization. Existing solutions towards Industry 4.0 either do not cover these areas completely or they address them partially. Therefore, even by directly combining current methods, a solution may not be achieved that would answer the needs for SM. This section of the thesis briefly introduces the machine monitoring protocols, standards, and tools that are needed for the development of the proposed platform. In addition, industry best practices and research findings to address the needs for data collection, data exchange, and data analytics between systems are discussed. Finally, a comparison is performed that describes advantages and disadvantages of the relevant approaches, which defined the goals of the desired framework.

2.1 SENSORS AND PLATFORMS

2.1.1 Standalone Sensors

Sensors are the devices that detect and measure physical and environmental phenomena such as motion, heat, moisture, vibration, and light. Sensors usually provide the measured variable in the forms of analog and digital signals. Machines and platforms in current century are usually equipped with built-in sensors. Part of these machines provide the data collected from the sensors to the users/developers for further implementation and developments. However, in some cases also there is a need to implement a standalone

sensor to the machine and collect the data directly. These sensors depending on their functionality and manufacturing decisions provide the data differently. Some of these protocols are analog, digital, Pulse Width Modulation (PWM), Inter-Integrated Circuit (I2C), Serial Peripheral Interface (SPI), Universal Asynchronous Receiver-Transmitter (UART), and Controller Area Network (CAN). With these protocols data is transmitted with different speeds and frequencies. An ideal IoT platform, therefore, needs to have the capability of communicating with all these protocols and be capable of acquiring data with the required frequency.

2.1.2 Programmable Logic Controllers

One of the common platforms used in the manufacturing industries are programmable logic controllers (PLCs). These platforms are usually capable of simple and low frequency I/O in Analog/Digital. Although these controllers are commonly used in the industries, they only address applications that require simple tasks. The limitations of PLCs such as lack of support in IoT communication protocols and low computing power make them limited in addressing the current needs required for the Industry 4.0 and SM.

2.1.3 Microcontrollers

Microcontroller unit, known as MCU, is a small computer made on a single integrated circuit [6]. Compared to PLCs, MCUs work with lower voltage as their electronics logic. In contrast with PLCs, microcontrollers are powerful and capable of communicating with many protocols and frequencies in the order of mega and gigahertz. Due to this compatibility and processing power, MCUs can communicate with a wide

variety of sensors, actuators, and networks, especially Internet, which make them an ideal solution for IoT applications.

2.1.4 Computer Numerical Control

Integration of computers with machine tools and automating them, opened a new generation of equipment, which are known as Computer Numerical Control (CNC) machines. These machines are controlled with a pre-programmed sequence of positions controlled with a computer. With the nature of CNC machines, they already have many sensors built-in that are required for the operation of the equipment. Some of the CNC machines provide these data for monitoring, optimization, and further developments. The challenging part involved in this process is the variety of standards, naming, and communication protocols used by different CNC manufacturers to provide the data.

2.2 COMMON COMMUNICATION FORMATS AND PROTOCOLS

2.2.1 RS-232 / TTL UART

One of the common standard protocols for transferring data in serial is RS-232 that communicates with logic voltage of smaller than 24 VDC. This standard is usually used in computer serial ports and consists of two data wires, one for transmitting and one for receiving the data. The speed of communication is configurable with the baud rate on both sides of the communication enabling hundreds of kilohertz of information transferring. The universal asynchronous receiver transmitter (UART), which is sometimes referred to as transistor-transistor logic (TTL), is another serial communication protocol that is widely used, especially in the microcontrollers. The logic voltage for UART is usually 5VDC,

which provides a reliable data transfer for short distances. The differences between UART and RS232 are mainly the logic voltage, opposite logic, and communication range. Figure 1 demonstrates the logic difference between these two protocols.

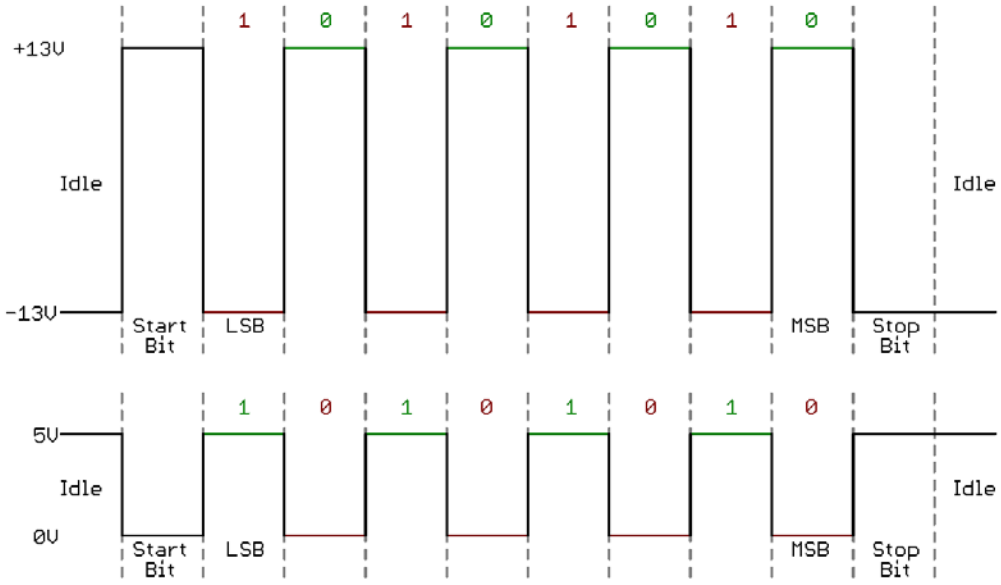


Figure 1. RS-232 (top) vs. TTL (bottom) for transferring 0b01010101 [7]

2.2.2 RS-485

Another common serial communication protocol is the RS-485, which is quite similar to RS-232, meaning that it communicates with two wires, one for transmitting and one for receiving data. The major difference of this protocol compared to RS-232 is the number of derivers and receivers on one line, which enables one driver active at a time to communicate to up to 32 devices. This protocol, also, can communicate in longer ranges than RS232. Table 1 below shows the differences between these two protocols in a table.

Table 1. Comparison table for RS232 and RS485 from RS485.com

SPECIFICATIONS		RS232	RS485
Mode of Operation		SINGLE ENDED	DIFFERENTIAL
Total Number of Drivers and Receivers on One Line (One driver active at a time for RS485 networks)		1 DRIVER	32 DRIVER
		1 RECVR	32 RECVR
Maximum Cable Length		50 FT.	4000 FT.
Maximum Data Rate (40ft. - 4000ft. for RS422/RS485)		20kb/s	10Mb/s-100Kb/s
Maximum Driver Output Voltage		+/-25V	-7V to +12V
Driver Output Signal Level (Loaded Min.)	Loaded	+/-5V to +/-15V	+/-1.5V
Driver Output Signal Level (Unloaded Max)	Unloaded	+/-25V	+/-6V
Driver Load Impedance (Ohms)		3k to 7k	54
Max. Driver Current in High Z State	Power On	N/A	+/-100uA
Max. Driver Current in High Z State	Power Off	+/-6mA @ +/-2v	+/-100uA
Slew Rate (Max.)		30V/uS	N/A
Receiver Input Voltage Range		+/-15V	-7V to +12V
Receiver Input Sensitivity		+/-3V	+/-200mV
Receiver Input Resistance (Ohms), (1 Standard Load for RS485)		3k to 7k	>=12k

2.2.3 Bluetooth and Wi-Fi

The need for communicating wirelessly led to development of communication protocols such as Bluetooth and Wi-Fi. Both of these protocols are compatible with 2.4GHz whereas Wi-Fi also supports 3.6 and 5GHz. The major difference in terms of data transfer between these two protocols is that Wi-Fi is capable of communicating with longer ranges and transferring data with higher bandwidth than Bluetooth. However, Bluetooth is still

utilized in some applications due to being much more power efficient [8-11]. As a result, Bluetooth is usually used for local applications where low data transfer is needed and Wi-Fi is commonly used for majority of wireless communications.

2.2.4 TCP and UDP

Transmission Control Protocol/Internet Protocol, referred to as TCP/IP, and User Datagram Protocol or Universal Datagram Protocol, referred to UDP, are a set of rules and procedures for the Internet communications [12]. TCP is a connection-oriented and confirms the delivery of the packets whereas UDP is a connectionless protocol and send the packets is the end of the relationship. Due to the structure of TCP, it communicates only in unicast, but UDP can communicate in unicast, multicast and broadcast. Therefore, TCP is suited when there is a need for high reliability of data delivery and time is not crucial. UDP, on the other hand, suits the applications where quick and efficient data transmission is needed [12-20]. Table 2 demonstrates a comparison between the TCP and UDP and provides a few example of the programs using these protocols [21].

Table 2. TCP vs UDP Protocols [21]

TCP	**UDP**
Keeps track of lost packets. Makes sure that lost packets are re-sent.	Doesn't keep track of lost packets
Adds sequence numbers to packets and reorders any packets that arrive in the wrong order	Doesn't care about packet arrival order
Slower, because of all added additional functionality	Faster, because it lacks any extra features
Requires more computer resources	Requires less computer resources
Examples of programs and services that use TCP: HTTP HTTPS FTP Many computer games	Examples of programs and services that use UDP: DNS IP telephony DHCP Many computer games

2.2.5 XML / JSON

Extensible Mark-up Language (XML) and JavaScript Object Notation (JSON) are two formats that use texts that are both human-readable and machine-readable to transfer data, mainly across the web. Although XML is designed to focus on representing the documents, it is now widely used also for representing arbitrary data structures. JSON objects, on the other hand, demonstrate attribute-value pairs and serialized data such as array data types. This format due to the simplicity of the use and structure is a very common format that is used as a replacement for the XML. Figure 2 from Google trend showing the usage of XML vs JSON APIs in the recent years. The following figure also, shows a set of data formatted into both JSON and XML formats.

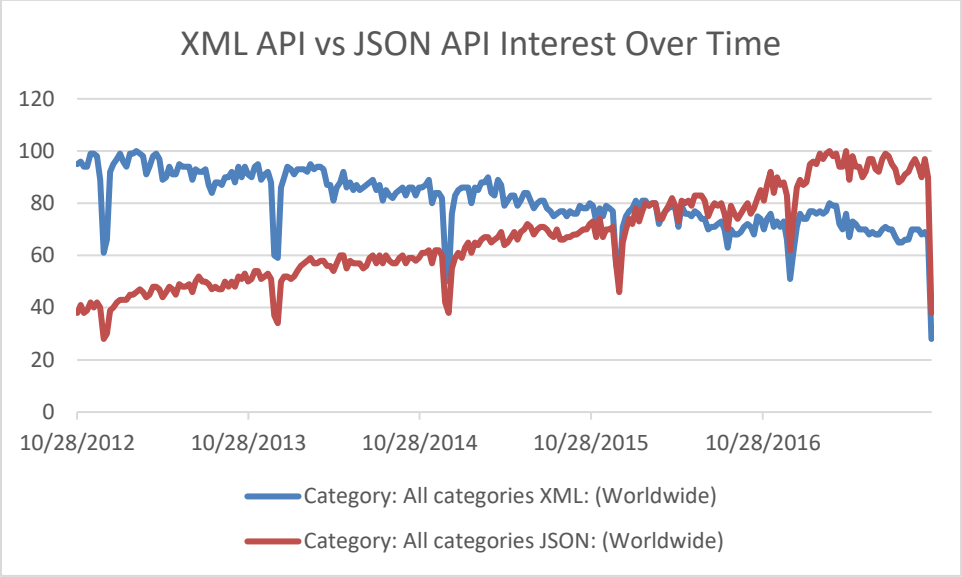


Figure 2. Google trends chart on XML API vs JSON API

http://localhost:8080/Json/SyncReply/Contacts

```
{
  - Contacts: [
      - {
            FirstName: "Demis",
            LastName: "Bellot",
            Email: "demis.bellot@gmail.com"
        },
      - {
            FirstName: "Steve",
            LastName: "Jobs",
            Email: "steve@apple.com"
        },
      - {
            FirstName: "Steve",
            LastName: "Ballmer",
            Email: "steve@microsoft.com"
        },
      - {
            FirstName: "Eric",
            LastName: "Schmidt",
            Email: "eric@google.com"
        },
      - {
            FirstName: "Larry",
            LastName: "Ellison",
            Email: "larry@oracle.com"
        }
    ]
}
```

http://localhost:8080/Xml/SyncReply/Contacts

```
<ContactsResponse xmlns:i="http://www.w3.org/20
  <Contacts>
    <Contact>
      <Email>demis.bellot@gmail.com</Email>
      <FirstName>Demis</FirstName>
      <LastName>Bellot</LastName>
    </Contact>
    <Contact>
      <Email>steve@apple.com</Email>
      <FirstName>Steve</FirstName>
      <LastName>Jobs</LastName>
    </Contact>
    <Contact>
      <Email>steve@microsoft.com</Email>
      <FirstName>Steve</FirstName>
      <LastName>Ballmer</LastName>
    </Contact>
    <Contact>
      <Email>eric@google.com</Email>
      <FirstName>Eric</FirstName>
      <LastName>Schmidt</LastName>
    </Contact>
    <Contact>
      <Email>larry@oracle.com</Email>
      <FirstName>Larry</FirstName>
      <LastName>Ellison</LastName>
    </Contact>
  </Contacts>
</ContactsResponse>
```

Figure 3. JSON vs XML formats

2.2.6 HTTP / REST

Currently, the majority of the Internet applications are based on Representational State Transfer (REST) architecture on Hypertext Transfer Protocol (HTTP) [22]. In other words, REST is a technical description of working principals behind the World Wide Web, known as web [23]. As discussed in the last section, one of the example applications of TCP is HTTP, which enables reliable communication of request-reply between two computers by tracking the lost packets and resending the data. Web browsers, web servers, automatic page downloaders, and many other applications use this protocol for their Internet

communication. The HTTP protocol uses the GET, POST, PUT, and DELETE commands to exchange/remove data between the client and server as shown in Figure 4.

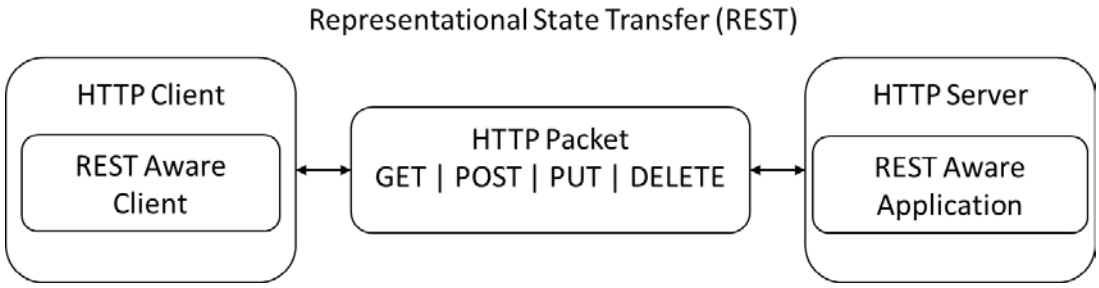


Figure 4. HTTP Protocol over REST Architecture

2.2.7 Publish Subscribe / MQTT

In contrast to REST architecture that the communication is done point-to-point, the publish/subscribe protocols are known as many-to-many and designed to transfer the data to many applications at the same time. There are many standardized messaging protocols that allow publish/subscribe communications. Some of the most popular ones are Data Distribution Service (DDS), Advanced Message Queuing Protocol (AMQP), Message Queue Telemetry Transport (MQTT), and Simple/Streaming Text Oriented Messaging Protocol (STOMP), and Extensible Messaging and Presence Protocol (XMPP). Depending on the type of the application, each one of these protocols or the combination of them can be utilized. As shown in Figure 5, the DDS provides the highest level of capabilities and securities. However, the other protocols such as MQTT, due to the light structure and less power consumption, are utilized for IoT applications and embedded systems.

	DDS	AMQP	CoAP	MQTT	REST / HTTP	XMPP
TRANSPORT	UDP/IP (unicast + multicast) TCP/IP	TCP/IP	UDP/IP	TCP/IP	TCP/IP	TCP/IP
INTERACTION MODEL	Publish-and-Subscribe, Request-Reply	Point-to-Point Message Exchange	Request-Reply (REST)	Publish-and-Subscribe	Request-Reply	Point-to-Point Message Exchange
SCOPE	Device-to-Device Device-to-Cloud Cloud-to-Cloud	Device-to-Device Device-to-Cloud Cloud-to-Cloud	Device-to-Device	Device-to-Cloud Cloud-to-Cloud	Device-to-Cloud Cloud-to-Cloud	Device-to-Cloud Cloud-to-Cloud
AUTOMATIC DISCOVERY	✓	-	✓	-	-	-
CONTENT AWARENESS	Content-based Routing Queries	-	-	-	-	-
QoS	Extensive (20+)	Limited	Limited	Limited	-	-
INTEROPERABILITY LEVEL	Semantic	Structural	Semantic	Foundational	Semantic	Structural
SECURITY	TLS, DTLS, DDS Security	TLS + SASL	DTLS	TLS	HTTPS	TLS + SASL
DATA PRIORITIZATION	Transport Priorities	-	-	-	-	-
FAULT TOLERANCE	Decentralized	Implementation-Specific	Decentralized	Broker is SPoF	Server is SPoF	Server is SPoF

Figure 5. Comparison of Messaging Protocols [24]

2.2.8 IFTTT

If-This-Then-That (IFTTT) is a free web service that allows creating conditional statements and interacting web services to one another. Changes that occur within web services such as Gmail, Facebook, or Instagram can trigger a service on IFTTT that triggers another web-app to perform an action such as sending an email.

2.2.9 MTConnect

With the need of collecting the data from the built-in sensors and existing information on the machines in a standard format, the development of MTConnect protocol was initiated [25]. MTConnect is an open-source, royalty-free, read-only, and XML-based protocol that allows data collection from manufacturing machines [26, 27]. MTConnect

consists of two sections of adapter and agent. The adapter retrieves the data from the machine and provides the data to the agent. The agent hosts the data in a TCP server and provides a REST interface, meaning that the data can be retrieved via HTTP request-reply. Therefore, multiple users can get access to the real-time data that are collected by the machine.

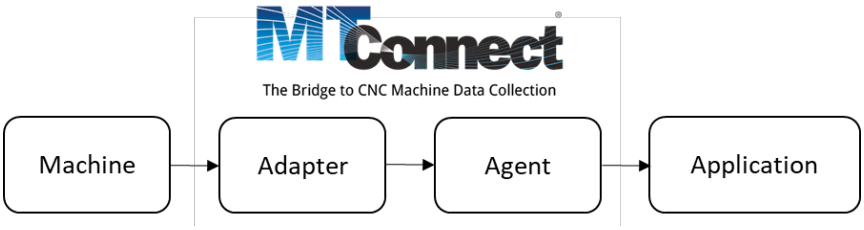


Figure 6. MTConnect Architecture

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="/styles/Streams.xsl"?>
<MTConnectStreams xmlns:m="urn:mtconnect.org:MTConnectStreams:1.3"
xmlns="urn:mtconnect.org:MTConnectStreams:1.3"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:mtconnect.org:MTConnectStreams:1.3
/schemas/MTConnectStreams_1.3.xsd">
  <Header creationTime="2017-10-28T02:06:30Z" sender="mtcagent"
instanceId="1490681234" version="1.3.0.9" bufferSize="131072"
nextSequence="4211083132" firstSequence="4210952060" lastSequence="4211083131"/>
  <Streams>
    <DeviceStream name="VMC-3Axis" uuid="000">
      <ComponentStream component="Rotary" name="C" componentId="c1">
        <Samples>
          <SpindleSpeed dataItemId="c2" timestamp="2017-10-28T02:04:45.206151"
name="Sspeed" sequence="4211054701" subType="ACTUAL">3400.0000000000</SpindleSpeed>
          <SpindleSpeed dataItemId="c3" timestamp="2017-10-28T01:56:24.500678"
name="Sovr" sequence="4210919524" subType="OVERRIDE">100.0000000000</SpindleSpeed>
```

Figure 7. Sample MTConnect XML data from agent.mtconnect.org/current

2.3 LIMITATIONS OF THE EXISTING SOLUTIONS

A few years after the initial development of web, several methods and frameworks were proposed to exchange the data from the machines and applications [28]. GE Fanuc

Automation has developed Cimplicity that allows monitoring of the factory and operations with an XML based data through WebView Screen. This service also is capable of generating and displaying alerts [29]. Another application called FactoryFlow, is developed by UGS that enables off-line access to the data of the factory-floor process, planning, and simulation. With the goal of accessing the CNC machines data from the internet, MDSI has developed an application called OpenCNC. This application is a software-only machine tool control that installs on a Windows PC to provide real-time data collection and publishing to a database. This allows the users to access OpenCNC's "Significant Events file", that is produced from collected PLC data [28]. Development of customized applications and interfaces such as Cimplicity, WebView, FactoryFlow, and OpenCNC, led to initiatives on development of an open communication standard to enable Internet connectivity to manufacturing equipment [30]. As a result, MTConnect as a standard protocol was recently developed that provides the data in only one direction, enabling the user to only read the data in real-time. Since the MTConnect communicates using the REST structure, as mentioned in the last sections, all internet-connected devices can get access to the data using the MTConnect with HTTP commands. Although MTConnect provides many benefits as a standard, it also has several limitations, which are discussed as follows:

2.3.1 MTConnect compatible CNC control should not be connected to the Internet

MTConnect is a service that is installed on a Windows-based CNC Control. Therefore, the Windows that is controlling the commands and interpretations of the codes to control the machine would also run the MTConnect and host the data as a webserver that could be accessed with a computer in the network. To expand the network and provide accessibility

to the data from the Internet, therefore, the control PC needs to be connected to the Internet with a static IP address. The security of the equipment could be increased by integration of firewalls and limited accessibility; however, there would be still a risk of internet attack as the packets are blocked using an application. Therefore, the CNC control itself is not recommended to be directly connected to the internet due to the security issues, which result in not having access to the MTConnect data.

2.3.2 Static IP address is required to access the MTConnect over the Internet

The other major limitation of the MTConnect is the fact that the data is hosted on a TCP webserver on the computer, meaning that in order to get access to the data a request needs to be send and wait for the response. As a result, when connecting the machine to the internet, the machine's IP address needs to be static so that the clients be able to send requests and receive the desired information. This problem has a direct relation with the fact that to get the data from the MTConnect, data needs to be pulled out and the MTConnect itself does not push the data out to the web.

2.3.3 There is no communication between the machines with MTConnect

A major area in Industry 4.0 is the capability of machine-to-machine communication. MTConnect, however, does not have the ability of communicating with one another and reading the information from the other equipment. This is due to the nature of the data exchange in the architecture of this protocol as the adapter is only able to communicate and provide the local variables of the equipment. Therefore, there is a lack of capability as a standalone protocol and use of another service would needed.

2.3.4 MTConnect requires an extensive infrastructure

Another major challenge in MTConnect, is complexity of the architecture and implementation. Development of the adapter and agent of the MTConnect for a non-compatible platform is very extensive and sometimes impossible for some embedded systems. With this structure, after collecting the data, it needs to be hosted instead of being transferred. Hosting the data requires development of Current and Probe pages of MTConnect and implementation of the TCP server that responds back to incoming requests. Therefore, there are a lot that need to be covered on the backend to produce the requirements of this protocol.

2.3.5 MTConnect is not suited for IoT

Currently, the MTConnect only delivers the data in XML format. XML is a web format, human-readable, and machine-readable. However, this format is heavy in size and very complicated to parse, which make the development and integration of IoT applications challenging. On the other hand, JSON not only is human and machine-readable, but also light-weight and very simple to parse. Since most of the web apps and APIs communicate in JSON, the MTConnect therefore, would not be a suitable option for integration of these services.

2.3.6 MTConnect does not provide historical data or perform analytics

Majority of the analytics require a historical set of data. MTConnect, however, due to its structure, only provides the latest received data. Since no buffer or storage dedicated in this protocol, it cannot perform analytics and display processed data. In addition to

historical analytics, there is also a lack of real-time data analytics, which is required for generating warnings and errors.

2.3.7 It is complicated to add new sensors to the equipment via MTConnect

This protocol is designed to collect the information from built-in sensors, meaning that it only provides the data already available on the equipment. Adding new sensors to the protocol, therefore, would be complicated and almost impossible in some cases since the sensor's data needs to be accessed by the controller, adapter, and agent. Additionally, since the data acquisition in MTConnect is in the range of Hertz, sensors that require high frequency data acquisition such as accelerometers cannot be implemented on this protocol.

2.4 GOALS OF THIS THESIS

Considering the infrastructure required for Industry 4.0, and the limitations of MTConnect as the latest standard for data exchange in manufacturing, this thesis presents a secure and cost-effective approach for data collection, data exchange, data storage, and cloud integration for MTConnect and non-MTConnect compatible platforms. More specifically, the proposed platform can be integrated into manufacturing equipment to: enable users to get access to the collected data from machines or standalone sensors from Internet with no need of having a static IP address; to receive data in JSON format in both request-reply and publish-subscribe protocols; to collect and store data locally and perform fog computing; and to integrate web services and web APIs to store and perform analytics on the cloud.

3. PROPOSED FRAMEWORK

This section presents the proposed approach that would address many of the limitations discussed earlier. In other words, this platform has the following capabilities:

- Compatible with MTConnect and Non-MTConnect sensors and devices
- Communicates with Internet and cloud without the need of static IP address
- Hardware secured between LAN and IAN
- Enables the machines and platforms to directly communicate with one another
- Capable of high frequency data acquisition from standalone sensors
- Facilitates fog computing and cloud computing

The contents in this section are categorized in two levels of hardware and software. The hardware section shows the general architecture of the platform and explains the structure of data exchange within the platform, and the software portion details on the protocols used and algorithms developed for collecting, processing, and exchanging the data to the cloud.

3.1 HARDWARE

The main objective of the proposed platform is to share the MTConnect data securely in a JSON format to the Internet services. Since MTConnect is mainly installed on a Windows operating system that is also the CNC control of the machine tool, direct Internet connectivity is not recommended due to the security issues. Although implementing firewalls on the PC may decrease the risk of Internet attacks; however, there would be possibilities of attacking the firewall and passing through it. High-end firewalls and secured routers are also very expensive and not cost effective. To address this problem, the

developed platform provides two independent networks that locally collects the data form the machine using MTConnect protocol and securely transfers the data to the Internet Area Network using one-directional UART communication as shown in Figure 8Figure 8. As discussed in the introduction, UART is a serial two-way communication protocol where incoming data and outgoing data are on two separate wires. The architecture proposed in this study only transfers the data from MCU1 to MCU2 and there is no physical possibility for the MCU2 to communicate with MCU1 due to the hardware design architecture. As a result, local and Internet area networks are entirely independent in hardware and the machine is secured from any activities happening in the IAN.

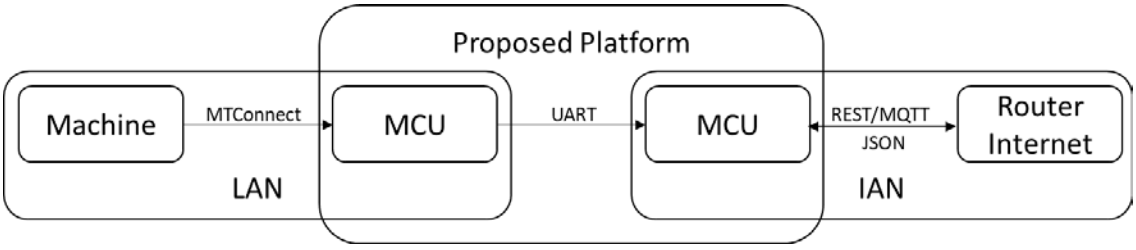


Figure 8. General architecture of the proposed approach

3.2 SOFTWARE

The software of this platform consists of the algorithms developed for embedded systems and the web applications created on the cloud. These can be categories into three main sub-sections of the codes on the first microcontroller for LAN, codes on the second microcontroller for IAN, and codes deployed on the cloud for the cloud communication and computations.

3.2.1 LAN MCU

The first microcontroller is responsible to perform the activities in the LAN, meaning that its main task is to collect the data from MTConnect using HTTP request-reply protocol. This microcontroller, therefore, needs to connect to the equipment either directly or via a switch/router. In case that the CNC machine is connected to a router, the microcontroller can be connected to the network with an Ethernet cable or with Wi-Fi. Due to the higher use of wired connection in shop floors, however, Ethernet connection was chosen in the sample implementation in this study. This microcontroller needs to request the data from the MTConnect and wait for the data to be received back. The incoming data in XML format is in the form of streaming, meaning that the data can be read and processed while coming to the system, or alternatively, it can be read thoroughly, then parsed, and processed. The process could be faster and more efficient in case of parsing and processing while reading the streaming data. However, in this case the data is no longer available for further use. Therefore, this study suggests that the data to be read and then processed. After reading the XML data of the MTConnect, the next task of this microcontroller is to detect the variables that contain a value. The pairs of variable-values should then be used to generate a lightweight JSON. Therefore, not only the data will be IoT friendly and compatible with many web-APIs, it will be compressed significantly. In addition to MTConnect, this microcontroller can also collect data from sensitive sensors where they should not be connected to the Internet. Regular sensors and information, however, are recommended to be connected to the second microcontroller for faster data acquisition and to reduce the amount of load on this MCU. After collecting the data, analytics can be performed on the data using the same microcontroller before transferring the data to the

Internet, which referred to as fog computing. An algorithm designed and implemented in this study with the goal of detecting and calculating the name, cycle time, and number of produced parts. The first part of this algorithm is to detect the spindle running cycle times. This can be done by starting a timer on the rising edge of the spindle speed, when it goes from zero to a certain speed, and reading the timer value on the falling edge, when the spindle speeds drops back to zero. Another parameter can be collected from MTConnect in addition to spindle speed, is the program name running on the machine, which is the same name as the G-Code file of that program. Correlating the program name and the spindle speed, the cycle times and number of parts could be identified as shown in Figure 9.

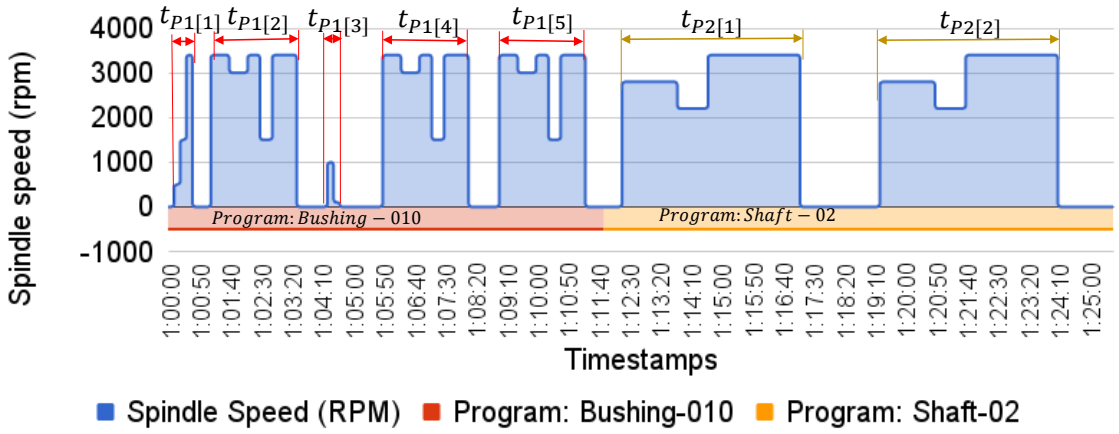


Figure 9. Analysis on spindle speed and program name to compute cycle time and number of produced parts

As shown in Figure 9, not all spindle cycle times correspond to the cycle times of the production times. Therefore, the algorithm initially categorizes the spindle cycles with their corresponding program names based on the timestamps. In this example, $t_{p1(i)}$ are considered for the Bushing program, and $t_{p2(i)}$ are assigned for Shaft program. For each

program name, the maximum spindle cycle time is identified and the cycle times less than half of this time, $t_{p1(1)}$ and $t_{p1(3)}$ in this example, are skipped out of the algorithm. This is done to filter out the small amounts where the spindle was used for setting up the machine. The remaining cycle times are used to calculate the mean and standard deviation of the production cycle times. The created JSON data and spindle cycle times combined with the program names are then transmitted to the second microcontroller using UART communication in only one direction.

3.2.2 IAN MCU

The second microcontroller is responsible for collecting the data from the first microcontroller and secondary sensors, process the data, and make them available for the Internet. This MCU, therefore, needs to be connected to the Internet while enabling UART communication with the first MCU and data collection from other sensors. Since there will be different types of data coming from the UART, and this microcontroller has no possible way of transferring data to the first microcontroller, the data needs to be marked by special characters and parsed in this microcontroller. While reading the incoming serial data, this MCU needs collect data from sensors and process them. Therefore, the tasks of this microcontroller need to be written in several threads, allowing them to work in parallel. Like the first MCU, this MCU can also perform fog computing on the data. For instance, this MCU can easily perform Fast Fourier Transform (FFT) on the data acquiring from an accelerometer and transfer the key frequencies to the Internet. With the current infrastructure with the Internet, the data cannot be directly transferred using Internet and needs to be fog computed. Moreover, this helps to reduce the traffic load on the network. Two communication protocols of REST and MQTT are recommended for this platform to

exchange data using the Internet network. As discussed, the instruction, REST is widely used and transfers the data in request-reply form from one computer to other. This communication can be utilized for all HTTP needs such as acquiring data from the websites, sending and receiving data from database, integrating third party web APIs, and more. In addition to REST, a publish-subscribe communication protocol is needed for IoT applications. This helps for easier and faster communication of the devices to one another and data exchange in the forms of many-to-many, meaning that multiple devices can read the data at once. There are many different publish-subscribe protocols, where from those Particle Cloud, Google PubSub, and generic MQTT are recommended due to simplicity of the structure. Therefore, data can be published and retrieved by multiple devices, and microcontrollers can communicate with one another over the Internet with no need for static IP address. Figure 10 shows the architecture of the communication between the microcontrollers connected to the Internet.

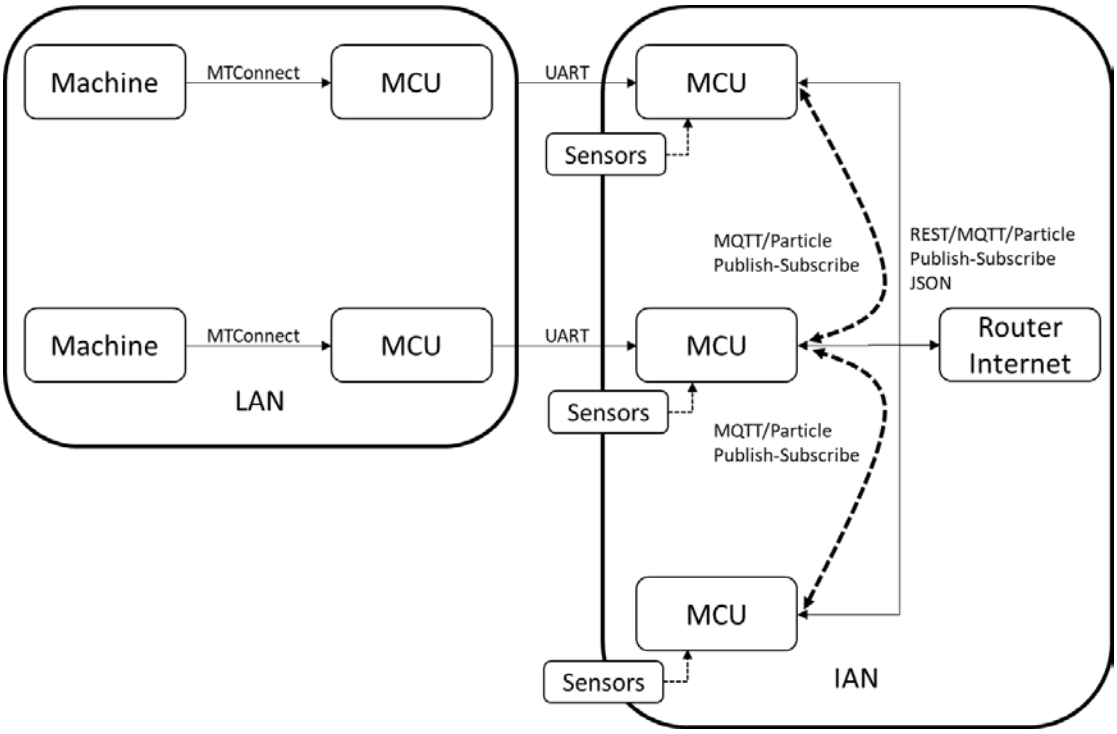


Figure 10. Architecture of the communication between second MCUs and Internet

3.2.3 Cloud Integration

Data collected on the second microcontroller can be retrieved via REST communication or the microcontroller itself can publish the data either using REST or a publish-subscribe protocol. These communication capabilities enable the platform to communicate with the web apps directly with no need of a secondary computer. The data, therefore, can be pushed directly to a web API, to insert the data to a database, for instance. Other applications that do not accept incoming data, can also pull the data out of the platform. For instance, to integrate Amazon Alexa to respond back a message from an equipment needs to use the REST communication and receive the data by sending a request. This platform, therefore, enables such services to collect the information they need from the equipment. As one of the critical capabilities needed for an equipment is the ability

to send alerting system, integration of REST with APIs such as Google Scripts, Twilio, and IFTTT allows the platform to send text messages, email, push notifications, and emergency calls. Moreover, this connectivity facilitates cloud computing and use of high performance computing systems for performing advanced analytics. For this study, an algorithm developed using Google Scripts to calculate the machine utilization based on the spindle usage.

4. IMPLEMENTATION AND DISCUSSION

To demonstrate an implementation of the proposed framework discussed in the last chapter, a platform was created and tested. The first part of this chapter describes the setup of the experiment and the conditions in which the platform was tested. The second part of the chapter describes the data collection procedure and fog computing implemented in the local area network. The last section presents the Internet connectivity and the results of the implementations on the cloud.

4.1 Experimental Setup

For this study, three CNC machines were considered. First machine is a HAAS VMC-3Axis, a milling CNC machine available online on the MTConnect website. This machine can be accessed anywhere from Internet with the following link: agent.mtconnect.org/current. The MTConnect data from this machine was used for the development and testing purposes. The second and third CNC machines in this study are two CNC lathes belonging to Georgia Institute of Technology, Okuma Multus and Okuma Genos, located in two different shop floors that are one mile away from each other.

MTConnect was already installed on these machines prior to this study. To collect the MTConnect data from these machines, Internet connectivity was needed; however, the machines should have been kept locally to keep them safe from cyber-attacks. These machines can communicate MTConnect through Ethernet RJ45 connectors, and the Internet was accessible throughout the shop floor with Wi-Fi. Therefore, for each machine, an Arduino Due with 84 MHz 32-bit ARM core microcontroller with Ethernet Shield was chosen for MTConnect data collection from the machines, and a Particle Photon with 120 MHz 32-bit Arm core microcontroller was chosen for the IAN to connect to the Internet using Wi-Fi. Figure 11 shows the architecture of this setup.

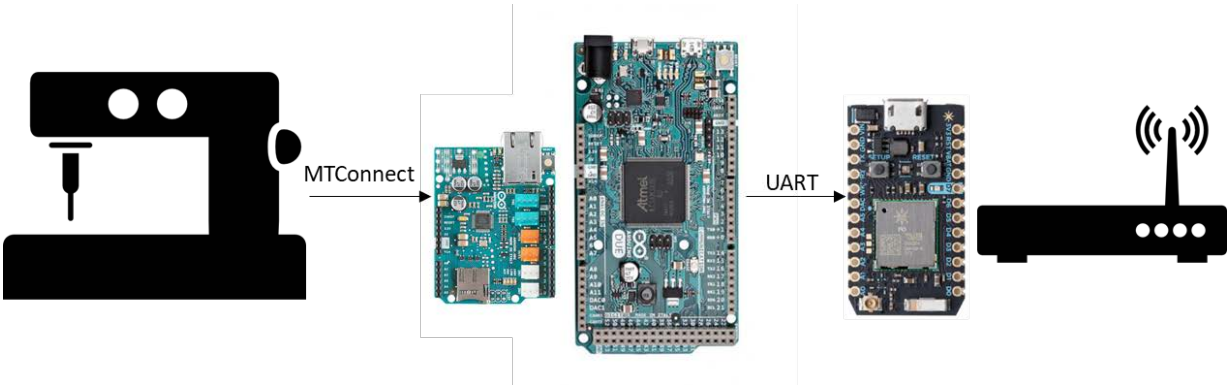


Figure 11. Use of Arduino Due and Particle Photon for the proposed framework

As shown in Figure 12, the only wires between the first and second microcontrollers are the power supply and the write-only UART data transfer from first to second microcontroller. In addition to collecting MTConnect data, an analog accelerometer, ADXL001, capable of detecting up to 500G was connected to the grove connectors on the second microcontroller, Particle Photon, to detect the shocks on the machine.

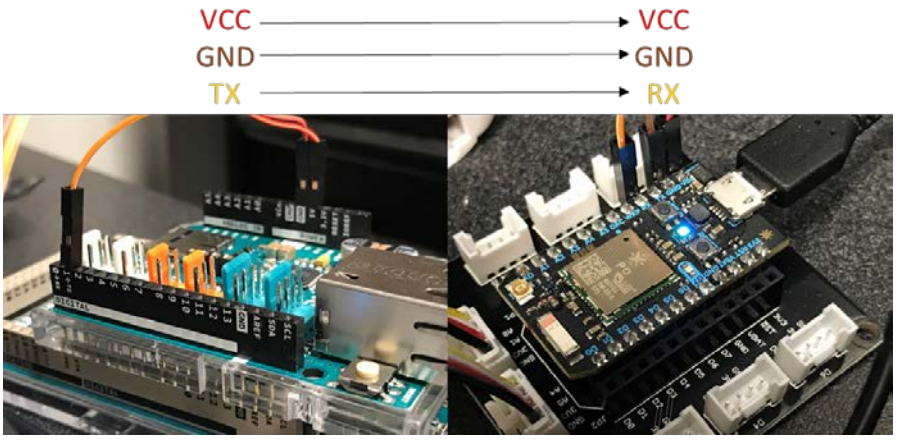


Figure 12. One-directional UART Connection from LAN to IAN

4.2 Data Collection Procedure and Fog Computing

First task of the LAN microcontroller, Arduino Due, is to collect the MTConnect data. Using the Ethernet shield the data from the VMC-3Axis machine from agent.mtconnect.org was collected. Due to the length of the MTConnect data, a small portion of it shown in Figure 13. Using the developed algorithm, this data is then efficiently parsed to the pairs of variable-values and the corresponding JSON array created and shown in Figure 14.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="/styles/Streams.xsl"?>
<MTConnectStreams xmlns:m="urn:mtconnect.org:MTConnectStreams:1.3"
xmlns="urn:mtconnect.org:MTConnectStreams:1.3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="urn:mtconnect.org:MTConnectStreams:1.3
/schemas/MTConnectStreams_1.3.xsd">
  <Header creationTime="2017-10-28T00:15:40Z" sender="mtcagent" instanceId="1490681234"
version="1.3.0.9" bufferSize="131072" nextSequence="4209551113" firstSequence="4209420041"
lastSequence="4209551112"/>
  <Streams>
    <DeviceStream name="VMC-3Axis" uuid="000">
      <ComponentStream component="Rotary" name="C" componentId="c1">
        <Samples>
          <SpindleSpeed dataItemId="c2" timestamp="2017-10-28T00:15:08.619294" name="Sspeed"
sequence="4209542074" subType="ACTUAL">3400.0000000000</SpindleSpeed>
          <SpindleSpeed dataItemId="c3" timestamp="2017-10-28T00:01:53.906009" name="Sovr"
sequence="4209352462" subType="OVERRIDE">100.0000000000</SpindleSpeed>
          <Load dataItemId="cl3" timestamp="2017-03-28T06:07:14.731817Z" name="Cload"
```

Figure 13. Portion of the MTConnect data collected using Arduino Due

```
{"c2":"3400.0000000000","c3":"100.0000000000","cl3":"UNAVAILABLE","cm":"SPINDLE","estop":"ARMED"
,"msg":"UNAVAILABLE","avail":"AVAILABLE","dev_asset_chg":"UNAVAILABLE","dev_asset_rem":"UNAVAILA
BLE","p2":"ON","Fovr":"100.0000000000","Frt":"0.3700589","Ppos":"UNAVAILABLE","cn2":"X0.089716
Y-
0.209607","cn3":"AUTOMATIC","cn4":"893","cn5":"FLANGE_CAM.NGC","cn6":"ACTIVE","cnt1":"UNAVAILABL
E","n3":"UNAVAILABLE","x2":"0.1011353433","x3":"0.0955354037","y2":"-0.2052114606","y3":"-
0.2073511863","y4":"UNAVAILABLE","z2":"-0.1000000015","z3":"-0.1000000000","z4":"UNAVAILABLE"}
```

Figure 14. Converted MTConnect data to JSON in Arduino Due

As discussed in the Proposed Framework, analytics can be performed on the collected data at this stage, which referred to as fog computing. The developed algorithm was used to detect the spindle running cycle times, which was done by starting a timer on the rising edge of the spindle speed, and reading the timer value on the falling edge, when the spindle speeds goes back to zero. Corresponding the spindle cycle times with the program names from MTConnect a string created where the first element is the program name and the next elements are the cycle times separated by pipe-bar characters. Each set of program-cycles is also separated by comma. This string was merged with the created MTConnect JSON, and separated with the special characters. The result is shown in Figure 15.

```
^
{"c2":"3400.0000000000","c3":"100.0000000000","cl3":"UNAVAILABLE","cm":"SPINDLE","es
top":"ARMED","msg":"UNAVAILABLE","avail":"AVAILABLE","dev_asset_chg":"UNAVAILABLE","
dev_asset_rem":"UNAVAILABLE","p2":"ON","Fovr":"100.0000000000","Frt":"0.3700589","Pp
os":"UNAVAILABLE","cn2":"X0.089716 Y-
0.209607","cn3":"AUTOMATIC","cn4":"893","cn5":"FLANGE_CAM.NGC","cn6":"ACTIVE","cnt1"
:"UNAVAILABLE","n3":"UNAVAILABLE","x2":"0.1011353433","x3":"0.0955354037","y2":"-
0.2052114606","y3":"-0.2073511863","y4":"UNAVAILABLE","z2":"-0.1000000015","z3":"-
0.1000000000","z4":"UNAVAILABLE"}~FLANGE_C|236343|193190|44256|359005|,@
```


Figure 15. Created string containing MTConnect-JSON and program-cycle times

This string is transferred using UART to the second microcontroller, Particle Photon. Using a thread dedicated for the incoming data, data is read and parsed. The spindle cycle times are then filtered using the algorithm defined in the Proposed Framework, shown in Figure 16, and the number of parts and their cycle time are computed. A sample calculation is shown in Figure 17.

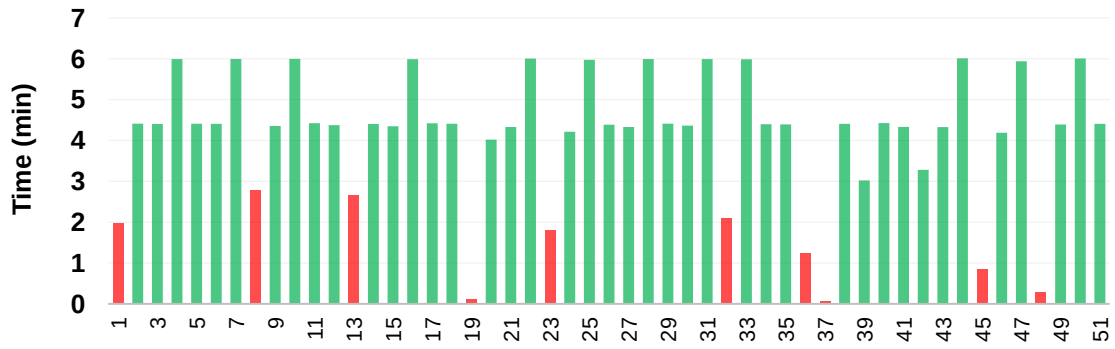


Figure 16. Spindle cycle filtering with the proposed algorithm

```
************* Spindle Cycles: ***************
BEARIN~1.TXT|1080000|1100000|8000|,FLANGE~1.TXT|3000000|3500000|2900000|2700000|,SHAFTN~1.TXT|1600000|
*****************************************
values: 1080000 1100000 8000
Maximum value: 1100000
numberOfValuesOfCycle considering tresholdCyclePercentage: 2
(valuesOfCycles[i] - averageCycleTime)^2: (1080000 - 1090000)^2 = 100000000.000000
(valuesOfCycles[i] - averageCycleTime)^2: (1100000 - 1090000)^2 = 100000000.000000
Sigma squares: 200000000.000000
valuesOfCycles: 1080000 1100000
averageCycleTime: 1090000
SDCycleTime: 10000

************************************************************
Program: BEARIN~1.TXT
Number of parts made: 2
Average cycle time (min): 18.166666
Standard deviation (min): 0.166667
Generated JSON:
{'program'='BEARIN~1.TXT', 'numberOfParts'='2', 'averageCycleTimeInMin'='18.166666', 'standardDeviatio
************************************************************

values: 3000000 3500000 2900000 2700000
Maximum value: 3500000
numberOfValuesOfCycle considering tresholdCyclePercentage: 4
(valuesOfCycles[i] - averageCycleTime)^2: (3000000 - 3025000)^2 = 625000000.000000
(valuesOfCycles[i] - averageCycleTime)^2: (3500000 - 3025000)^2 = 2286698496.000000
(valuesOfCycles[i] - averageCycleTime)^2: (2900000 - 3025000)^2 = 2740098048.000000
(valuesOfCycles[i] - averageCycleTime)^2: (2700000 - 3025000)^2 = 2545786880.000000
Sigma squares: 3902603264.000000
valuesOfCycles: 3000000 3500000 2900000 2700000
averageCycleTime: 3025000
SDCycleTime: 294745

************************************************************
Program: FLANGE~1.TXT
Number of parts made: 4
Average cycle time (min): 50.416668
Standard deviation (min): 4.912427
Generated JSON:
{'program'='FLANGE~1.TXT', 'numberOfParts'='4', 'averageCycleTimeInMin'='50.416668', 'standardDeviatio
************************************************************
```

Figure 17. Sample cycle time analysis done by Particle Photon

4.3 Cloud Integration and Result

Internet is a common method of communication and data exchange, which eases the accessibility and scalability. As the largest network in the world, Internet allows connected devices to interact with one another regardless of where they are located. Every day the number of IoT platforms and companies providing cloud services such as storage, analytics, and web services are increasing [31]. Integration of the Internet and cloud functionalities to this platform, therefore, allows unlimited possibilities. Some of the main services are demonstrated such as pushing data to a database, request-respond and publish-subscribe of data, cloud computation, web app integration, and alerting systems.

4.3.1 Pushing data to cloud database

In addition to making the data available for the Internet services to pull the data, with this approach the data can be directly pushed to a database without the need of a secondary computer. Figure 18 shows an example of inserting a value, spindle speed in this case, to a database on the Google Cloud Platform. Due to the integration of publish/subscribe protocols of Particle API and MQTT, platforms can communicate with one another over the Internet or with cloud services such as AWS IoT, Lambda, and RDS as shown in Figure 19.

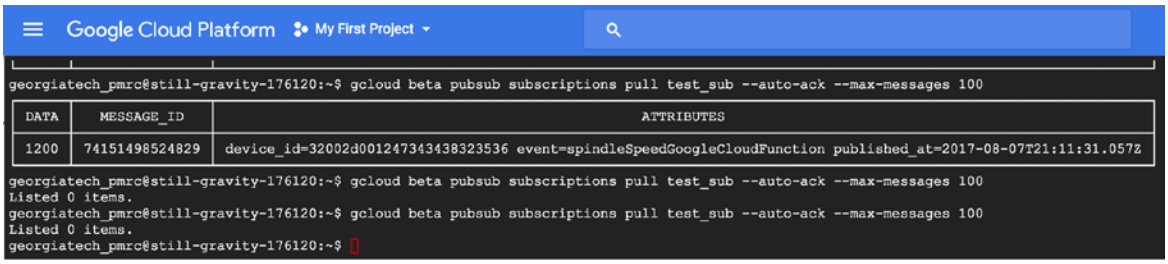


Figure 18. Spindle speed pushed to a database in Google Cloud

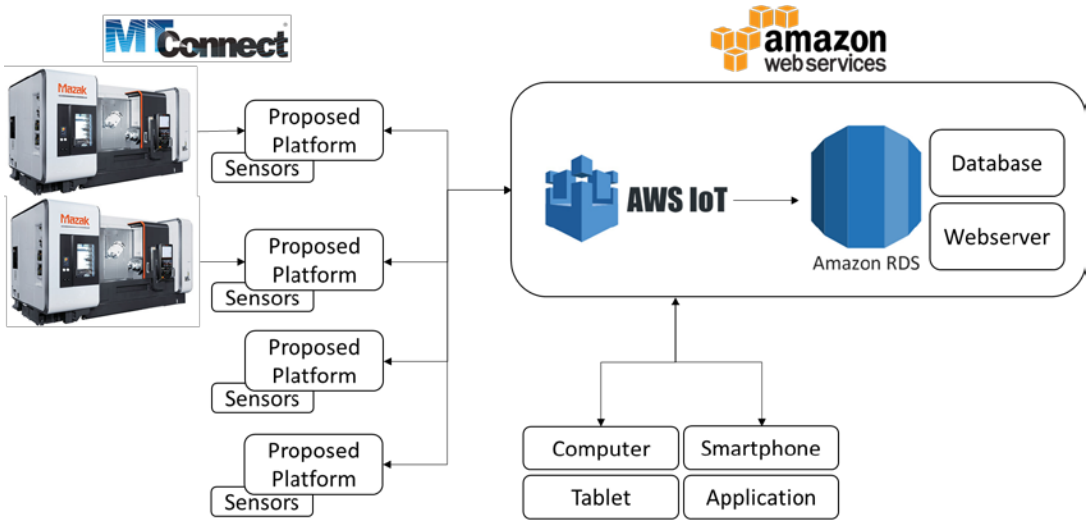


Figure 19. Framework of data exchange with multiple platforms to AWS

4.3.2 Request-reply and publish-subscribe protocols for data transfer

As mentioned in previous chapters, REST/HTTP protocol in this platform facilitates the request-reply data exchange and publish-subscribe is done using Particle API and MQTT protocol. Data defined as cloud variables can be read by sending an HTTP request with the format below where the device identification and access token are required and unique per platform:

https://api.particle.io/v1/devices/[DeviceId]/[Variable]?access_token=[AccessToken]

Since different CNC machines and platforms that are compatible with MTConnect name their variables differently, the developed platform renames the variables to a uniform naming to standardize the developed API. As a result, both original and standard versions of the MTConnect data can be collected from this platform. To access this data with the REST protocol, an HTTP request with the variable names of MTCOriginal and MTCStandard can be sent to the API. The results of these two variables are shown in Figure 20 side by side for comparison.


```
{"cmd":"VarReturn",
"name":"MTCStandard",
"result":"
{"c2":"3400.0000000000",
"c3":"100.0000000000",
"cl3":"UNAVAILABLE",
"cm":"SPINDLE",
"estop":"ARMED",
"msg":"UNAVAILABLE",
"avail":"AVAILABLE",
"dev_asset_chg":
"UNAVAILABLE",
"dev_asset_rem":"UNAVAILABLE",
"p2":"ON",
"Fovr":"100.0000000000",
"Frt":"0.4",
"Ppos":"UNAVAILABLE",
"cn2":"X-1.415634 Y-0.701698",
"cn3":"AUTOMATIC",
"cn4":"217",
"cn5":"FLANGE_CAM.NGC",
"cn6":"ACTIVE",
"cnt1":"UNAVAILABLE",
"n3":"UNAVAILABLE",
"x2":"-1.3923978806",
"x3":"-1.3956432983",
"y2":"-0.7444807887",
"y3":"-0.7385052624",
"y4":"UNAVAILABLE",
"z2":"-0.1000000015",
"z3":"-0.1000000000",
"z4":"UNAVAILABLE"}
","coreInfo":
{"last_app":"",
"last_heard":"2017-08-22T17:23:40.850Z",
"connected":true,
"last_handshake_at":"2017-08-22T17:16:49.957Z",
"deviceID":"32002d001247343438323536",
"product_id":6}}
```

```
{"cmd":"VarReturn",
"name":"MTCOriginal",
"result":"
{"spindleSpeed":"3400.0000000000",
"spindleOverride":"100.0000000000",
"load":"UNAVAILABLE",
"rotaryMode":"SPINDLE",
"emergencyStop":"ARMED",
"message":"UNAVAILABLE",
"availability":"AVAILABLE",
"assetChanged":"UNAVAILABLE",
"assetRemoved":"UNAVAILABLE",
"powerStatus":"ON",
"feedover":"100.0000000000",
"feedrate":"0.4",
"pathPosition":"UNAVAILABLE",
"block":"X-1.415634Y-.701698",
"controllerMode":"AUTOMATIC",
"line":"217",
"program":"FLANGE_CAM.NGC",
"execution":"ACTIVE",
"toolId":"UNAVAILABLE",
"loadX":"UNAVAILABLE",
"x2":"-1.3923978806",
"x3":"-1.3956432983",
"loadY":"-0.7444807887",
"positionActualY":"-0.7385052624",
"positionCommandedY":"UNAVAILABLE",
"loadZ":"-0.1000000015",
"positionActualZ":"-0.1000000000",
"positionCommandedZ":"UNAVAILABLE"}
","coreInfo":
{"last_app":"",
"last_heard":"2017-08-22T17:23:39.618Z",
"connected":true,
"last_handshake_at":"2017-08-22T17:16:49.957Z",
"deviceID":"32002d001247343438323536",
"product_id":6}}
```


Figure 20. HTTP response from MTCOriginal and MTCStandard variables

The same procedure can be used for this platform to retrieve the raw spindle utilization cycle times, processed spindle cycle times, and a voice response string created based on

the processed data for integration of Amazon Alexa. This information can also be published using the Particle API for publish/subscribe or with MQTT protocol as shown in Figure 21. The results of these requests are as follows:

- The spindle utilization cycles, which show the durations when the spindle speed was non-zero, were fog computed in the first microcontroller. Program names with their corresponding cycle utilizations can be accessed through the second microcontroller in IAN, using request-reply and publish-subscribe protocols. An example result is shown below:

  FLANGE_C|264246|264488|264332,DARRENYO|3461|1374706|38245|11187

- Similarly, the spindle utilization cycle times were fog computed in the second microcontroller to find out the production cycle time and the number of parts produced. This processed data can be accessed similar to the raw cycle times. The example below shows the result of this process in a JSON array:

  [{"program":"FLANGE_C","numberOfParts":"94","averageCycleTimeInMin":"4.85","standardDeviationInMin":"0.80"}r,{"program":"DARRENYO","numberOfParts":"5","averageCycleTimeInMin":"58.28","standardDeviationInMin":"23.12"},{"program":"BEARIN~1.TXT","numberOfParts":"2","averageCycleTimeInMin":"18.17","standardDeviationInMin":"0.17"},{"program":"FLANGE~1.TXT","numberOfParts":"4","averageCycleTimeInMin":"50.42","standardDeviationInMin":"4.91"},{"program":"SHAFTN~1.TXT","numberOfParts":"1","averageCycleTimeInMin":"26.67","standardDeviationInMin":"0.00"}]

- The example below also, shows an sample string created as the voice response for Amazon Alexa integration:

  VMC-3Axis with program name FLANGE_C has produced 94 pieces with average cycle time of 4.85 minutes, with program name DARRENYO has produced 5 pieces with average cycle time of 58.28 minutes, with program name BEARIN~1.TXT has produced 2 pieces with average cycle time of 18.17 minutes, with program name FLANGE~1.TXT has produced 4 pieces with average cycle time of 50.42 minutes, with program name SHAFTN~1.TXT has produced 1 pieces with average cycle time of 26.67 minutes,

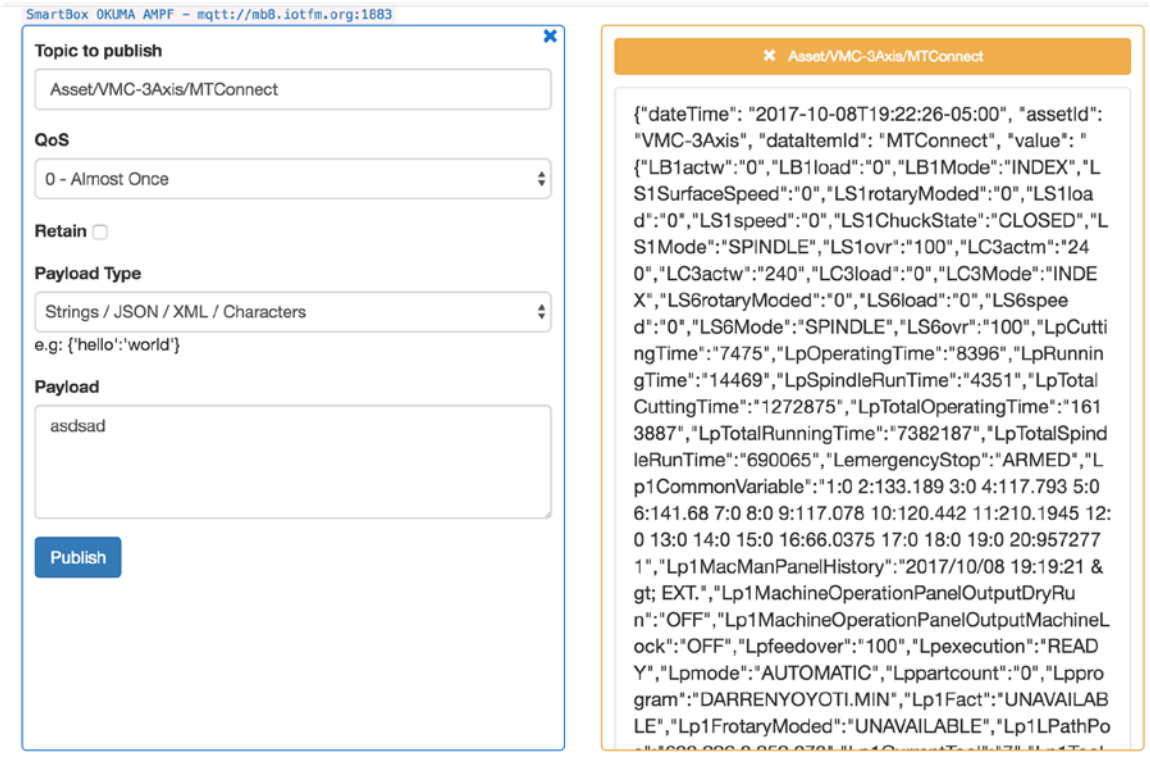


Figure 21. Incoming MTConnect data from MQTT message broker

4.3.3 Data exchange for web app development

IoT systems, no matter where they are located, only require an Internet connection to make the data available for the other applications. This allows the sensors and machines to be placed in any distance from one another. To visualize the data, therefore, a web application should be developed that collects the data from each one of the IoT devices, or the data could be accessed via a secondary data source such as a message broker or a database. Even though querying a database using a request-reply communication takes very small amount of time in order of milliseconds, receiving the data from a subscribed topic on message broker with a publish/subscribe method gives a more efficient communication for receiving real-time data. To access the historical data, a database needs to be implemented and the data should be retrieved from the database. However, for development of applications where only the real-time data is needed, publish/subscribing would eliminate querying the database repeatedly. As an example, to show the current/real-time values of an equipment on a webapp or to send a notification message, publish/subscribing would be an appropriate communication. To test this experiment on the developed platform, an accelerometer was connected to the second microcontroller and the accelerations more than a threshold were considered as shocks to the system.

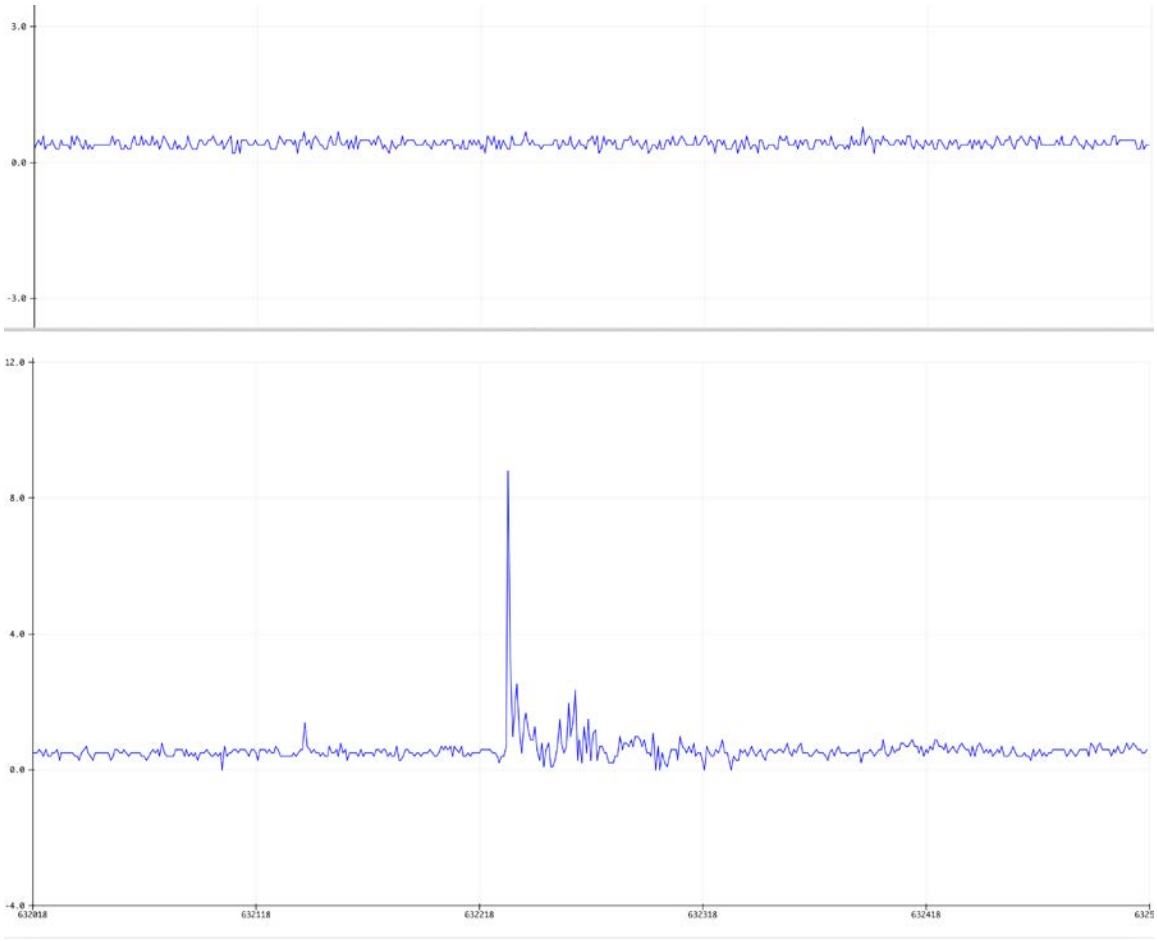


Figure 22. Actual data collected from accelerometer for shock monitoring, before and after hitting the machine

When detecting a shock to the system, the platform can send alerting messages to the operator or shop floor manager with the amount of the shock and severity of the accident. With the development of the alerting functions and integration of Google Scripts, Twilio, Particle, and IFTTT APIs, the platform could successfully send alerting emails, text messages, push notifications, and voice calls, respectively to defined personals as shown in Figure 23.

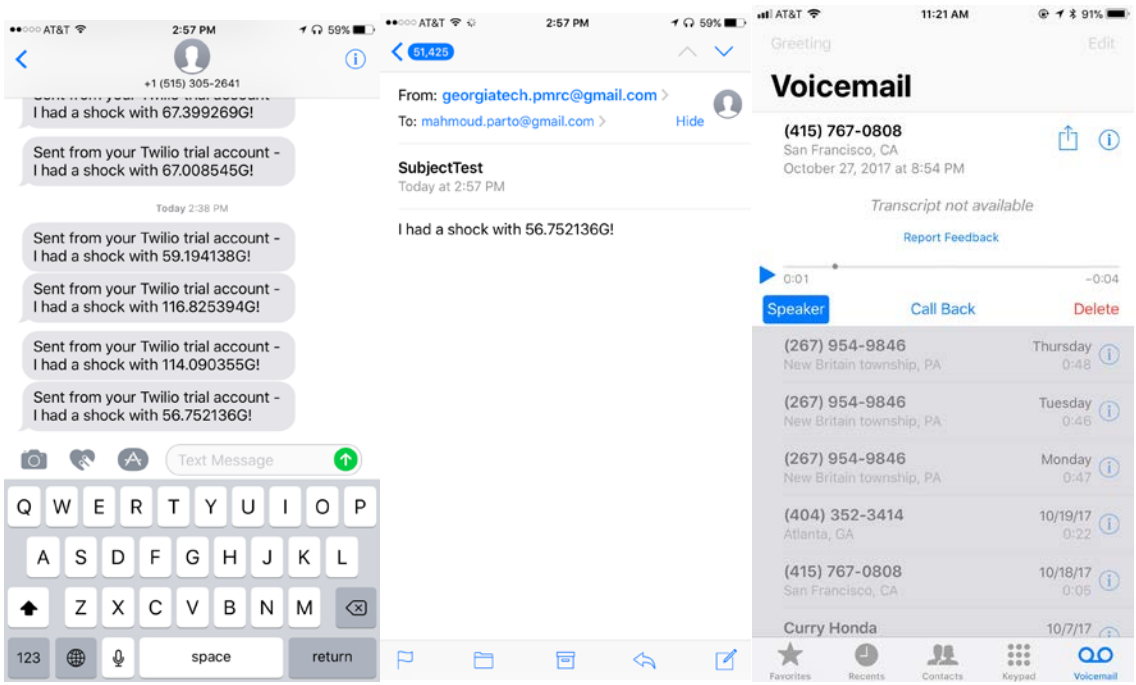


Figure 23. Alerting text message, email, and voicemail sent from the platform by integration of Twilio, Google Scripts, and IFTTT

Another web service that could be utilized by exchanging the data collected from the equipment is performing the computation and implementing the analytics algorithms on the cloud, known as cloud computing. Local computers have the limitations of accessibility, performance, storage. Cloud resources, on the other hand, are managed by companies dedicated for this work, they can be accessed from anywhere using an Internet connection, and their performance could be adjusted by sharing the process to multiple computing resources and their storage can be expanded based on the required needs. Moreover, cloud services are mostly cost efficient and some even come with no costs. Google applications and scripts, for instance, are free of cost for a limited storage, which could be sufficient for a variety of applications. In this study, an algorithm developed on the Google Scripts Web Apps that collects the data from the platform and computes the spindle utilization of the equipment. The results are the utilization for the last 100 minutes,

24 hours, 30 days, and 12 months, which can be accessed via REST communication to the web app. The equipment's real-time data, historical data, results of fog computing, and cloud computing, therefore, could all be accessed from a web app or smartphone app as shown in Figure 24 and Figure 25.

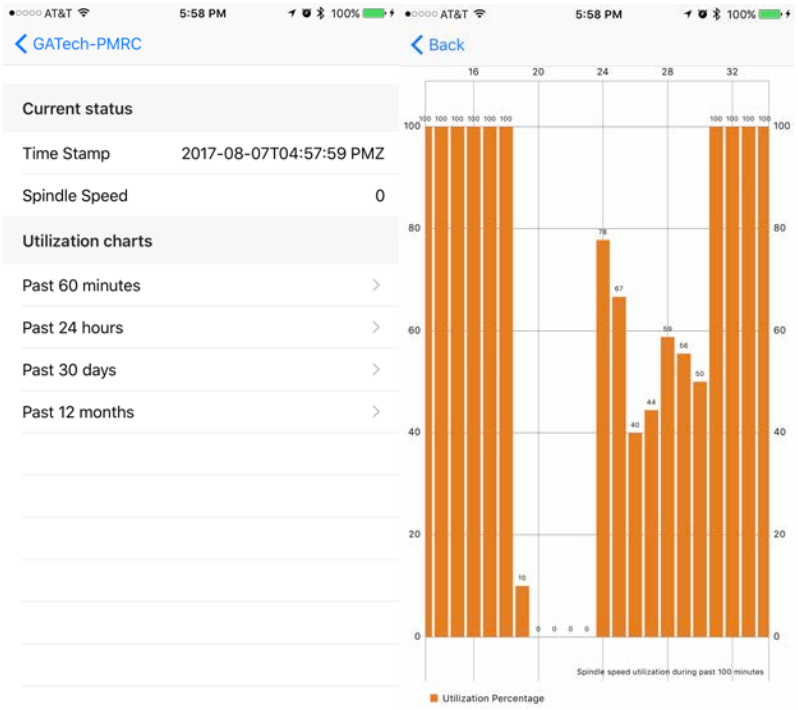
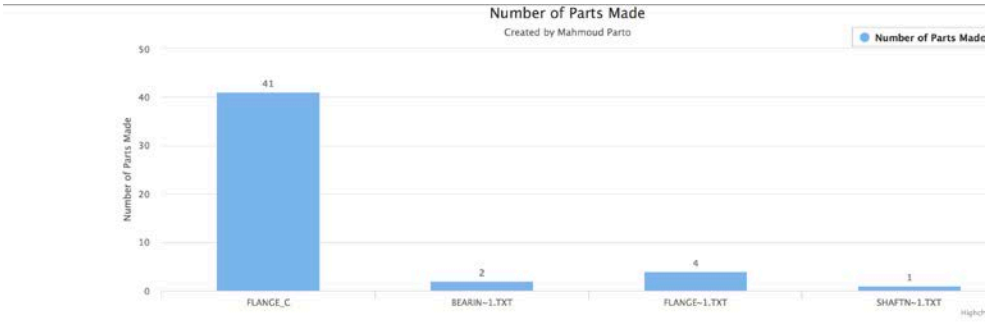
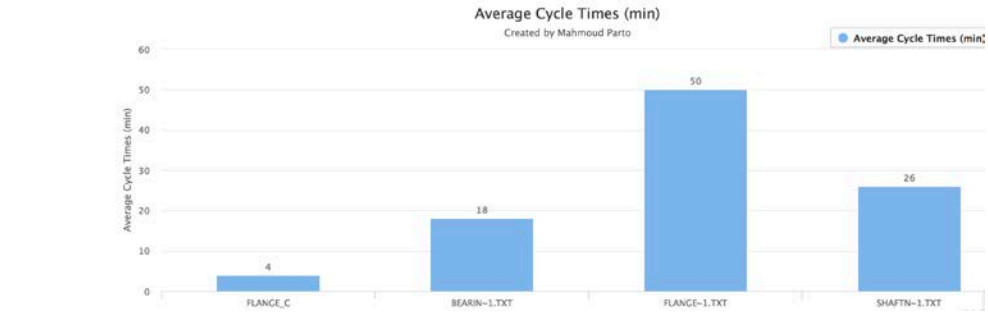


Figure 24. iOS app developed to visualize cloud computed machine utilization





program	numberOfParts	averageCycleTimeInMin	standardDeviationInMin
FLANGE_C	41	4.78	0.82
BEARIN~1.TXT	2	18.17	0.17
FLANGE~1.TXT	4	50.42	4.91
SHAFTN~1.TXT	1	26.67	0.00

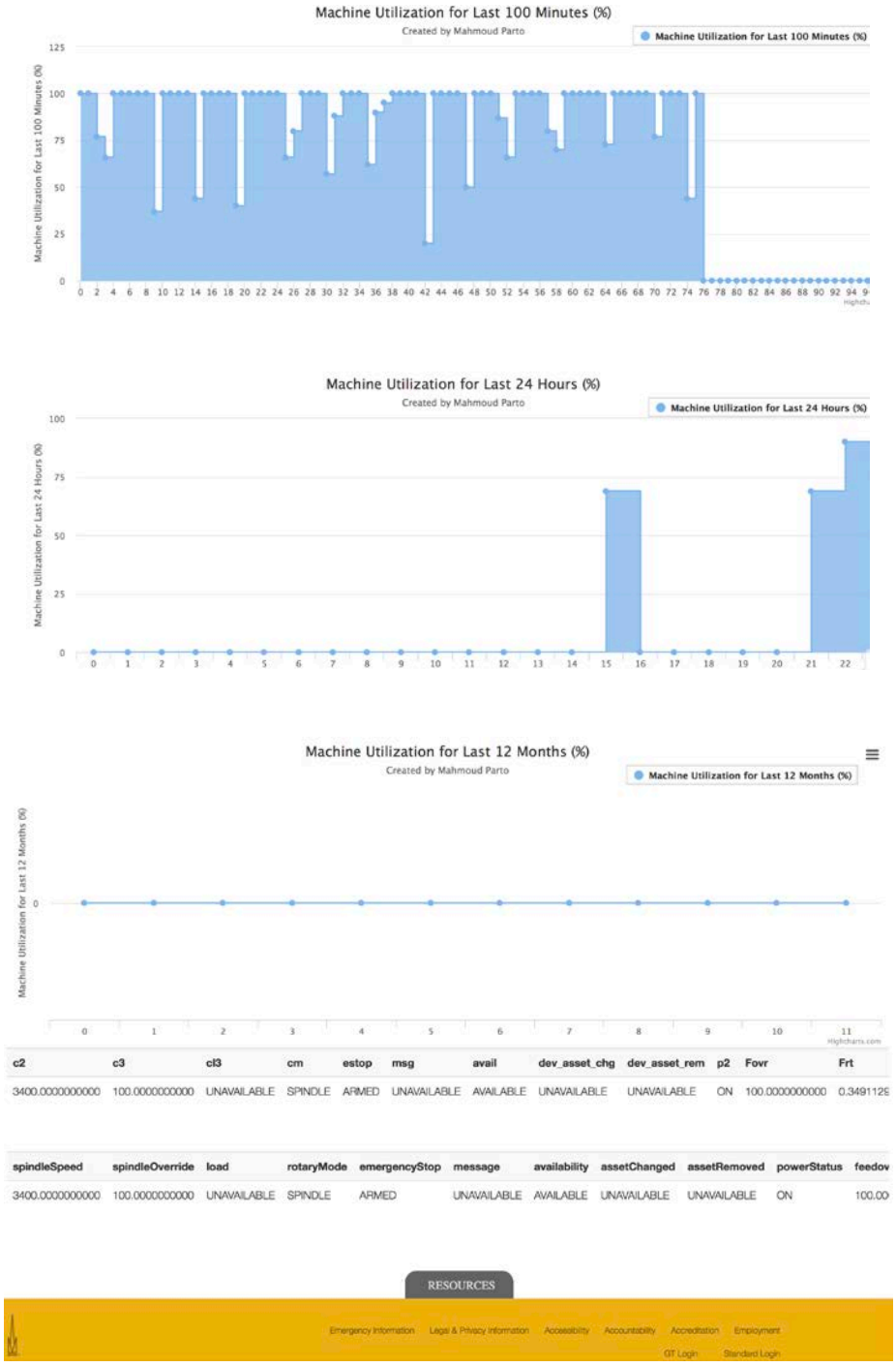


Figure 25. Web app developed to show the real-time data, historical data, results of fog computing, and cloud computing

As demonstrated in this section, integration of Internet accessibility and communicating in a compact and standard language via this platform address many of the limitations discussed earlier. In summary, the IAN portion of the created platform facilitates request-reply and publish-subscribe protocols to allow accessing MTConnect data in JSON format, transferring data to cloud databases, cloud computing for utilization and cycle time analytics, and integration of third party web services such as Amazon, IFTTT, Twilio, and Google, for integration of Alexa, alerting systems, text messaging, emailing, voice mailing. With this setup, the data on the IAN microcontroller could be collected as fast as 120MHz in independent mode or less if used in a multithreaded application while using the processing power for other tasks. The results of this experiment also show that the process of reading the MTConnect data with 84MHz, parsing the data, creating the corresponding JSON, transmitting the data to IAN with 250,000 baudrate, and publishing the data via MQTT to AWS, takes 4 seconds approximately.

5. CONTRIBUTION AND CONCLUSION

5.1 Contributions of this thesis

- This work presented an approach for development of a platform for collecting high frequency data from MTConnect and non-MTConnect sensors and platforms and securely sharing compressed data in JSON format to the cloud applications and services using request-reply and publish-subscribe protocols.
- The use of request-reply and publish-subscribe protocols enables the web applications to access the real-time and historical data with ease. It also facilitates

machines and platforms to communicate directly with one another in real-time over the Internet without the need of a static IP address or a secondary host.

- The security of the MTConnect data collection in this method is on the hardware level, which prevents any cyber-attacks from entering the equipment anyhow. More specifically, the proposed platform creates two separate networks of LAN and IAN that are connected in only one direction.
- Integration of various communication protocols in this platform, allows communication to web APIs which opens unlimited possibilities such as inserting data to a cloud database, providing data for web apps or smartphone apps, sending alerting messages, and integration of web services such as Google Clouds, Amazon Web Services, Amazon Lambda, If-This-Then-That (IFTTT), and many more.
- Scripts and algorithms developed in this thesis are attached in the appendix for further development. The majority of these scripts are written in C++ and JavaScript as libraries for microcontrollers and open-source cloud services to ease their use in other applications. These scripts include the following topics:
    - Reading MTConnect data over TCP
    - Converting MTConnect parameter-values to JSON
    - Fog computations on the spindle speed, to find out the program name and number of parts made and their cycle time
    - UART communication from LAN to IAN with special characters separation
    - Access to original and standardized MTConnect in JSON format for easy integration of web apps and APIs using REST, Particle API, and MQTT
    - Cloud computations of machine utilization

- iOS app for visualization of machine utilization results
- Machine-machine communication using Particle API and MQTT
- Amazon Alexa integration using Amazon Lambda and IFTTT
- Pushing data to a cloud database
- High frequency data acquisition using multithreaded application
- Web API integration for alerting system, text message with Twilio, emailing with Google Scripts, and sending voicemails using IFTTT

5.2 Limitations of the Study and Future Recommendations

- Hardware limitations: the proposed framework presents the use of two microcontrollers for data collection and cloud communication. This was done due to the I/O capability, high frequency data acquisition, cost efficiency, and simplicity of cloud integration for the platform. Improvements can be investigated by using computers and integration of them with embedded systems to increase the efficiency and rate of data exchange.
- Software limitations: in this thesis, structures and protocols of REST/HTTP for request-reply, and MQTT and Particle API for publish-subscribe communications were integrated, due to their popularity and simplicity of their structure. For future studies, cons and pros of the other protocols such as Data Distribution Service (DDS) can be investigated. DDS, in addition to the benefits of REST/HTTP and MQTT, provides UDP communication in addition to TCP, allows cloud-to-cloud in addition to device-to-device and device-to-cloud communications, and facilitates data prioritization. These capabilities are not considered or limited in the structure of the other protocols [32].

5.3 Conclusion

This thesis presented an approach for development of a platform for collecting data from MTConnect and non-MTConnect platforms, compressing the available values in the data to JSON format, and securely sharing the data to the cloud applications and services using request-reply and publish-subscribe protocols. Use of request-reply and publish-subscribe protocols allows the web apps to access the real-time and historical data. This also facilitates machines and platforms to communicate directly with one another in real-time over the Internet without the need of a secondary host and static IP address. The security of the MTConnect data collection in this method is on the hardware level, which prevents any cyber-attacks from entering the equipment. More specifically, the proposed platform creates two separate networks of LAN and IAN that are connected in only one direction. Integration of various communication protocols in this platform, allows communication to web APIs which opens unlimited possibilities such as inserting data to a cloud database, providing data for web apps or smartphone apps, sending alerting messages, and integration of web services such as Google Clouds, Amazon Web Services, Amazon Lambda, If-This-Then-That (IFTTT), and many more. To verify the outcomes of this approach, three platforms were created using Arduino Due and Particle Photon and tested on three CNC machines, VMC-3Axis, Okuma Genos, and Okuma Multus. The created platforms successfully showed the MTConnect data in standardized tag names in JSON format using both response-request and publish-subscribing protocols updating every four seconds; the spindle utilization cycles, number of parts and cycle times were fog and cloud computed; data was pushed to a cloud database; connectivity to AWS including Amazon Lambda for Alexa integration was successful; and the platform worked

properly with web APIs of Twilio, Google Scripts, Particle, and IFTTT. Finally, the software and hardware limitations of this study were discussed, and areas for future work and investigation to improve the limitations of this approach were suggested.

APPENDIX A. DESCRIPTION OF THE CODE AND ALGORITHM

A.1 Arduino Due Code to collect and parse MTConnect data, create JSON, perform the algorithm to find out spindle running cycle times, and transfer the data using UART

```
//  Created by Mahmoud Parto
//  Copyright Â© 2017. All rights reserved.

/*****************************************************************************************************************
  Section  : Function prototypes
*****************************************************************************************************************/
void startTheConnection();
void httpRequestFromMTC();
String parseWithDataItemId(String MTCString, String dataItemId);
String getTimeStampWithDataItemId(String MTCString, String dataItemId);
void overWriteInSDCard(String fileName, String text);
void appendInSDCard(String fileName, String text);
String readFromSDCard(String fileName);
//String readFromSDCardFromOpenedFile(File file);


/*****************************************************************************************************************
  Section  : Global variables
*****************************************************************************************************************/
String MTCResponse = "";
String valueString = "";
unsigned long timeMillisProgramStart;
unsigned int spindleSpeedValuePrev = 0;

/*****************************************************************************************************************
  Section  : Ethernet/URL Configuration
*****************************************************************************************************************/
#include <SPI.h>
//#include <Ethernet.h>
#include <Ethernet2.h>
EthernetClient client;

// Website details
const char server[] = "agent.mtconnect.org";
const String hostWebsite = "www.mtconnect.org";
```

```
const String subPage = "/current";
//const String subPage = "/sample?interval=0&heartbeat=1000"; // for streaming.. not implemented
completely
int port = 80;


//byte server[] = { 192, 168, 0, 15};
//int port = 5000;
//const String subPage = "/current";

// Set the static IP address to use if the DHCP fails to assign
IPAddress ip(192, 168, 1, 94);


/*****************************************************************************************************
************************************
  Section  : SD Card Configuration

*****************************************************************************************************
************************************/
//#include <SPI.h> //Already included
#include <SD.h>

/*****************************************************************************************************
************************************
  Section  : Setup

*****************************************************************************************************
************************************/
void setup()
{
 //************************************ Serial communication initialization
 Serial.begin(250000);
 Serial.println("Hello!");
 //************************************ SD Card initialization
 if (!SD.begin(4))
 {
  Serial.println("SD initialization failed!");
 }

 //************************************ Ethernet communication initialization
 startTheConnection();
 delay(1000);
 httpRequestFromMTC();
}



/*****************************************************************************************************
************************************
  Section  : Main Loop

*****************************************************************************************************
************************************/
void loop()
{
```

```
  // incoming data from HTTP response:
  while (client.available())
  {
     MTCResponse += client.readStringUntil('>');
     MTCResponse += String('>');
     if( MTCResponse.endsWith("</MTConnectDevices>"))
     {
       break;
     }
     if( MTCResponse.endsWith("</MTConnectStreams>"))
     {
       break;
     }
     delay(2); // allow bits to comes
  }
//  Serial.println(MTCResponse);
  // if the server's disconnected (finished reading data), restart the connection, and http request again:
  if (!client.connected())
  {
    String currentDataToSend = parseAndGetAllData(MTCResponse);

    String program = parseWithDataItemId(MTCResponse, "cn5"); //Lpprogram
    program = program.substring(0,8); // ONLY 8 character is allowed for the file name
    String spindleSpeed = parseWithDataItemId(MTCResponse, "c2"); //LS1speed

//    String emergencyStop = parseWithDataItemId(MTCResponse, "estop");
//    String powerState = parseWithDataItemId(MTCResponse, "p2");
//    String currentDataToSend =
//      "{'program':'" + String(program) + "'," +
//      "'spindleSpeed':'" + String(spindleSpeed) + "'," +
//      "'emergencyStop':'" + String(emergencyStop) + "'," +
//      "'powerState':'" + String(powerState) + "'}";

    // if we have a program loaded in the machine
    if (program != "")
    {
      unsigned int spindleSpeedValue = spindleSpeed.toInt();

      if (spindleSpeedValuePrev == 0)
      {
        // if spindle speed goes from 0 to a value (Starting Point)
        if(spindleSpeedValue > 0)
        {
          timeMillisProgramStart = millis();
//          Serial.println("Spindle went from 0 to a non-zero value");
        }
      }
      else
      {
        // if spindle speed goes from a value to 0 (Ending Point)
        if (spindleSpeedValue == 0)
        {
          unsigned long timeDifference = millis() - timeMillisProgramStart;
          appendInSDCard(program, String(timeDifference) + "|");         // disable it for now

//          Serial.println("Spindle went to 0");
```

```
        Serial.println("TimeDifference: " + String(timeDifference));
       }
      }
      spindleSpeedValuePrev = spindleSpeedValue;
    }


    Serial.print('^');
    //Serial.print(MTCResponse);
    Serial.print(currentDataToSend);
    Serial.print('~');


    File root = SD.open("/");
    root.rewindDirectory();
    //printDirectory(root, 0);
    while (true)
    {

      File file =  root.openNextFile();
      if (! file)
      {
        break;
      }
      if(!file.isDirectory())
      {
        String fileName = file.name();
        if(!fileName.startsWith("_"))
        {
          String text = readFromSDCardFromOpenedFile(file);
          Serial.print(fileName + "|" + text + ",");
        }
      }
      file.close();
    }
    root.close();



    // Send ending character for processed data
    Serial.print('@');
    Serial.println();

    MTCResponse = "";
    client.stop();
    startTheConnection();
    //Serial.println("HTTP requesting again...");
    httpRequestFromMTC();
  }
}

/*****************************************************************************************
************************************
  Function Name  : startTheConnection
  Description    : Starts the network connection with Ethernet to read MTConnect data from CNC machine
```

```
*****************************************************************************************
*************************************/
void startTheConnection()
{
//  Serial.println("starting the connection...");
  // Assign a mac address for the ethernet
  byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };

  // start the Ethernet connection:
//  if (Ethernet.begin(mac) == 0)
//  {
//    Serial.println("Failed to configure Ethernet using DHCP");
//    // try to congifure using IP address instead of DHCP:
    Ethernet.begin(mac, ip);
//  }
}


/*****************************************************************************************
*************************************
  Function Name  : httpRequestFromMTC
  Description    : Starts the network connection with Ethernet to read MTConnect data from CNC machine

*****************************************************************************************
*************************************/
void httpRequestFromMTC()
{
  if (client.connect(server, port))
  {
//    Serial.println("connected");
    // Make a HTTP request:
    client.println("GET " + subPage + " HTTP/1.1");
//    client.println("Host: " + hostWebsite);
    client.println("Connection: close");
    client.println();
  }
  else
  {
    Serial.println("connection failed");
  }
}


/*****************************************************************************************
*************************************
  Function Name  : parseWithDataItemId
  Description    : parse received MTConnect with given dataItemId

*****************************************************************************************
*************************************/
String parseWithDataItemId(String MTCString, String dataItemId)
{
  String value = "";
  const int MTCLength = MTCString.length();
  const int idLength = dataItemId.length();
```

```
  char MTCBuffer[MTCLength];
  char idBuffer[idLength];

  MTCString.toCharArray(MTCBuffer, MTCLength);
  dataItemId.toCharArray(idBuffer, idLength);

  String thisRangeWord = "";
  for (int i = 0; i < MTCLength - idLength; i++)
  {
    // Optimization:
    if (MTCBuffer[i] != idBuffer[0])
    {
      continue;
    }

    // find the word starting index i with length dataItemId
    thisRangeWord = "";
    for (int j = i; j < idLength + i; j++)
    {
      thisRangeWord += MTCBuffer[j];
    }

    // compare if this word is the same as dataItemId
    if (thisRangeWord == dataItemId)
    {
      // go until you reach '>'
      while (MTCBuffer[i++] != '>') {}

      // get the value until you reach <
      while (MTCBuffer[i++] != '<')
      {
        value += MTCBuffer[i-1];
      }
      //Serial.println(dataItemId + " :" + value);
      break;
    }
  }
  return value;
}


/**************************************************************************************************
************************************
  Function Name  : getTimeStampWithDataItemId
  Description    : parse received MTConnect to get time stamp of the given dataItemId

**************************************************************************************************
************************************/
String getTimeStampWithDataItemId(String MTCString, String dataItemId)
{
  String value = "";
  const int MTCLength = MTCString.length();
  const int idLength = dataItemId.length();
  char MTCBuffer[MTCLength];
  char idBuffer[idLength];
```

```
  MTCString.toCharArray(MTCBuffer, MTCLength);
  dataItemId.toCharArray(idBuffer, idLength);

  String thisRangeWord = "";
  for (int i = 0; i < MTCLength - idLength; i++)
  {
    // Optimization:
    if (MTCBuffer[i] != idBuffer[0])
    {
      continue;
    }

    // find the word starting index i with length dataItemId
    thisRangeWord = "";
    int j = i;
    for (j = i; j < idLength + i; j++)
    {
      thisRangeWord += MTCBuffer[j];
    }

    // compare if this word is the same as dataItemId
    if (thisRangeWord == dataItemId)
    {
      i = j+1;
      // go until you reach ""
      while (MTCBuffer[i++] != "") {}

      // get the value until you reach <
      while (MTCBuffer[i++] != "")
      {
        value += MTCBuffer[i-1];
      }
      //Serial.println(dataItemId + " :" + value);
      break;
    }
  }
  return value;
}


/*********************************************************************************************
************************************
   Function Name  : overWriteInSDCard
   Description    : Overwrite the given text to the text file (fileName) in SD card

*********************************************************************************************
************************************/
void overWriteInSDCard(String fileName, String text)
{
  File file;
  file = SD.open(fileName, O_WRITE | O_CREAT | O_TRUNC); // Overwrite
  if (file)
  {
    file.println(text);
  }
  else
```

```
  {
    Serial.println("Couldn't write to file");
  }
  file.close();
}

/*********************************************************************************************
************************************
   Function Name  : appendInSDCard
   Description    : Append the given text to the text file (fileName) in SD card

*********************************************************************************************
************************************/
void appendInSDCard(String fileName, String text)
{
//  Serial.println("Trying to write to file: " + fileName + " with this text: " + text);
  File file;
  file = SD.open(fileName, FILE_WRITE); // Append
  if (file)
  {
    Serial.println("File opened.");
    file.print(text);
  }
  else
  {
    Serial.println("Couldn't write to file");
  }
  file.close();
}

/*********************************************************************************************
************************************
   Function Name  : readFromSDCard
   Description    : Read the whole text in the text file (fileName) from SD card
   Input          : fileName: name of the text file with extension

*********************************************************************************************
************************************/
String readFromSDCard(String fileName)
{
  String tempString = "";
  File file;
  file = SD.open(fileName);
  if (file)
  {
    while (file.available())
    {
      tempString += char(file.read());
    }
  }
  else
  {
    Serial.println("Couldn't read from file");
  }
  file.close();
  return tempString;
```

```
}

/*******************************************************************************************
************************************
  Function Name  : readFromSDCard
  Description    : Read the whole text in the text file (fileName) from SD card
  Input          : file: the opened File

*******************************************************************************************
************************************/
String readFromSDCardFromOpenedFile(File file)
{
 String tempString = "";
 if (file)
 {
  while (file.available())
  {
   tempString += char(file.read());
  }
 }
 else
 {
  Serial.println("Couldn't read from file");
 }
 return tempString;
}



/*******************************************************************************************
************************************
  Function Name  : parseAndGetAllData
  Description    : parse received MTConnect with all of the dataItemIds and return JSON

*******************************************************************************************
************************************/
String parseAndGetAllData(String MTCString)
{
 String resultJSON = "{";
 String dataItemIdTag = "dataItemId";
 String dataItemId = "";
 String value;
 const int MTCLength = MTCString.length();
 const int idLength = dataItemIdTag.length();
 char MTCBuffer[MTCLength];
 char idBuffer[idLength];

 MTCString.toCharArray(MTCBuffer, MTCLength);
 dataItemIdTag.toCharArray(idBuffer, idLength);

 String thisRangeWord = "";
 for (int i = 0; i < MTCLength - idLength; i++)
 {
  // Optimization:
  if (MTCBuffer[i] != idBuffer[0])
  {
```

```
      continue;
    }

    // find the word starting index i with length dataItemIdTag
    thisRangeWord = "";
    for (int j = i; j < idLength + i; j++)
    {
      thisRangeWord += MTCBuffer[j];
    }

    // compare if this word is the same as dataItemIdTag
    if (thisRangeWord == dataItemIdTag)
    {
      /////// Get the name of the data (dataItemId)
      // go until you reach "
      while (MTCBuffer[i++] != '\"') {}
      // get the value until you reach "
      while (MTCBuffer[i++] != '\"')
      {
        dataItemId += MTCBuffer[i-1];
      }

      // go until you reach '>'
      while (MTCBuffer[i++] != '>') {}

      // get the value until you reach <
      while (MTCBuffer[i++] != '<')
      {
        value += MTCBuffer[i-1];
      }
      value.trim();

      if(value != "")
      {
        resultJSON += ("\"" + dataItemId + "\":\"" + value + "\",");
      }
      dataItemId = "";
      value = "";
//      Serial.println(dataItemIdTag + " :" + value);
    }
  }


  resultJSON += "}";
  // remove the last , at the end
  resultJSON.replace(",}" , "}");

  return resultJSON;
}
```

A.2 Code on Particle IDE to read the MTConnect data from UART

```
//  Created by Mahmoud Parto
```

```
//  Copyright Â© 2017. All rights reserved.

#include "application.h"
#include "readFromSerial.h"

/*******************************************************************************************************
*************************************
 * Function Name  : readWholeDataFromSerial
 * Description    : Reads the whole data received by serial1 (TX-RX)
 * Output Type    : String

*******************************************************************************************************
*************************************/
String readWholeDataFromSerial()
{
   String MTConnectString = "";
   MTConnectString = Serial1.readString();
   return MTConnectString;
}

/*******************************************************************************************************
*************************************
 * Function Name  : getDataFromMTConnect
 * Input          : MTConnect: incoming MTConnect string
 * Description    : Reads the whole MTConnect data from input MTConnect string
 * Output Type    : String

*******************************************************************************************************
*************************************/
String getDataFromMTConnect(String MTConnect)
{
   return splitBetweenTwoChars(MTConnect, '^', '~');
}


/*******************************************************************************************************
*************************************
 * Function Name  : getCyclesFromMTConnect
 * Input          : MTConnect: incoming MTConnect string
 * Description    : Reads the whole spindle cycles from input MTConnect string
 * Output Type    : String

*******************************************************************************************************
*************************************/
String getCyclesFromMTConnect(String MTConnect)
{
   return splitBetweenTwoChars(MTConnect, '~', '@');
}

/*******************************************************************************************************
*************************************
 * Function Name  : splitBetweenTwoChars
 * Input          : inputString, firstChar, secondChar
 * Description    : split the incoming string that is between two characters
 * Output Type    : String
```

```
**********************************************************************************************
**********************************/
String splitBetweenTwoChars(String inputString, char firstChar, char secondChar)
{
   String result = "";
   int firstIndex = inputString.indexOf(firstChar);
   int secondIndex = inputString.indexOf(secondChar);
   if(firstIndex > -1 && secondIndex > -1)
   {
      result = inputString.substring(firstIndex + 1,secondIndex);
   }
   return result;
}


// char buffer[20000];
// char c;
// long j;
// long i;
// bool isDataComing;
//
/**********************************************************************************************
**********************************
//  * Function Name  : readProccessedDataFromSerial
//  * Description    : Reads (ProgramName, Cycle periods when spindle was non zero) (until ~) received by
serial1 (TX-RX)
//                   This has to be called right after readMTConnectFromSerial due to the structure of the passed
data
//  * Output Type    : String
//  **********************************************************************************************
**********************************/
// String readSpindleCycleTimesFromSerial() // not used in this version
// {
//    return readUntil('@');
// }

//
/**********************************************************************************************
**********************************
//  * Function Name  : readUntil
//  * Description    : Reads Serial1 (from RX) until it reaches the check input char
//  * Output Type    : String
//  **********************************************************************************************
**********************************/
// String readUntil(char chr) // not used in this version
// {
//    // Serial.println("reading until: " + String(chr));
//    // clear the array
//    memset(buffer, 0, sizeof buffer);
//    Serial.begin(250000); // USB Serial
//    Serial1.begin(250000); // TX RX Serial

//    // read until you reach the input chr
//    i = 0;
//    j = 0;
```

```
//    while (c != chr)
//    {
//       if(Serial1.available())
//       {
//          i = 0;
//          c = Serial1.read();
//          if(chr == '@')
//          {
//             Serial.print(c);
//          }

//          if(c != chr)
//             buffer[j++] = c;
//       }
//       else // in case of not getting data, don't get stuck in while
//       {
//          i++;
//          if(i > 10000)
//          {
//             break;
//          }
//       }
//    }
//    // Serial.println(buffer);
//    return String(buffer);
// }
```

A.3 Code on Particle IDE for part number and cycle time analytics from spindle cycles

```
//  Created by Mahmoud Parto
//  Copyright Â© 2017. All rights reserved.

#include "spindleCycleAnalysis.h"
#include "math.h"
#include <ArduinoJson.h>

extern char alexaDataBuffer[2560];
String alexaString = "";
// first split the data with , and send them to processThisData
String processIncomingData(String incomingData)
{
   // Serial.println("Processing: " + incomingData);
   alexaString = "VMC-3Axis ";
   String tempString = "";
   String processedData = "";

   for(long i = 0; i < incomingData.length(); i++)
   {
      if(incomingData[i] != ',')
      {
         tempString += incomingData[i];
      }
      else
      {
         processedData += processThisData(tempString);
```

```
            if( i != incomingData.length() - 1)
               processedData += ",";
            tempString = "";
         }
      }
      // Serial.println("alexaString: " + alexaString);
      strcpy(alexaDataBuffer, alexaString.c_str());

      processedData = "[" + processedData + "]";
      return processedData;
}

String processThisData(String thisData)
{
      // Serial.println("Processing: " + thisData);
      String programName = "";
      int startingIndexOfValues = 0;
      // find out the number of values in thisData
      int numberOfValues = getNumberOfValues(thisData);
      // make an array of ints with count = numberOfValues
      long values[numberOfValues];
      int valueIndex = 0;
      float tresholdCyclePercentage = 0.5;

      // first get the program name from the first field
      char c;
      for(int i = 0; i < thisData.length(); i++)
      {
         c = thisData[i];
         if(c != '|')
         {
            programName += String(c);
         }
         else
         {
            startingIndexOfValues = i+1;
            break;
         }
      }

      // get the values from the next fields
      String thisValue;
      for(long i = startingIndexOfValues; i < thisData.length(); i++)
      {
         if(thisData[i] != '|')
         {
            thisValue += thisData[i];
         }
         else
         {
            values[valueIndex++] = atol(thisValue);
            thisValue = "";
         }
      }

      // // Print values:
```

```
    // // Serial.print("values: ");
    // for(int i = 0 ; i < numberOfValues; i++)
    // {
    //     Serial.print(String(values[i]) + " ");
    // }
    // Serial.println();

    // find the max of the values
    long maxValue = 0;
    for(int i = 0; i < numberOfValues; i++)
    {
       if(maxValue < values[i])
       {
          maxValue = values[i];
       }
    }
    // Serial.println("Maximum value: " + String(maxValue));

    // calculate the average, standard deviation, and number of parts from cycle times based on defined
tresholdCyclePercentage
    int numberOfValuesOfCycle = 0;
    // find the count of ValuesOfCycle
    for(int i = 0 ; i < numberOfValues; i++)
    {
       if(float(values[i])/float(maxValue) > tresholdCyclePercentage)
       {
          numberOfValuesOfCycle++;
       }
    }
    // Serial.println("numberOfValuesOfCycle considering tresholdCyclePercentage: " +
String(numberOfValuesOfCycle));

    // put the cycles in to the valuesOfCycles
    long valuesOfCycles[numberOfValuesOfCycle];
    // Serial.println("numberOfValuesOfCycle" + String(numberOfValuesOfCycle));
    // Serial.println("numberOfValues" + String(numberOfValues));
    int j = 0;
    for(int i = 0 ; i < numberOfValues; i++)
    {
       if(float(values[i])/float(maxValue) > tresholdCyclePercentage)
       {
          valuesOfCycles[j++] = values[i];
       }
    }

    // calculate the avearage of valuesOfCycles
    long averageCycleTime = 0;
    for(int i = 0; i < numberOfValuesOfCycle; i++)
    {
       averageCycleTime += valuesOfCycles[i];
    }
    averageCycleTime /= numberOfValuesOfCycle;

    // calculate the standard deviation
    float SDCycleTime = 0;
    for(int i = 0; i < numberOfValuesOfCycle; i++)
```

```
    {
        float temp = (float(valuesOfCycles[i]) - float(averageCycleTime))*(float(valuesOfCycles[i]) -
float(averageCycleTime));
        SDCycleTime += temp;
        // Serial.println("(valuesOfCycles[i] - averageCycleTime)^2: (" + String(valuesOfCycles[i]) + " - " +
String(averageCycleTime) + ")^2 = " + String(temp));
    }
    // Serial.println("Sigma squares: " + String(SDCycleTime));
    SDCycleTime /= float(numberOfValuesOfCycle);
    SDCycleTime = sqrt(SDCycleTime);


    // Serial.print("valuesOfCycles: ");
    // for(int i = 0 ; i < numberOfValuesOfCycle; i++)
    // {
    //     Serial.print(String(valuesOfCycles[i]) + " ");
    // }
    // Serial.println();
    // Serial.println("averageCycleTime: " + String(averageCycleTime));
    // Serial.println("SDCycleTime: " + String(long(SDCycleTime)));
    // Serial.println();

    int numberOfParts = numberOfValuesOfCycle;
    float averageCycleTimeInMin = averageCycleTime/1000.0/60.0;
    float SDinMIn = SDCycleTime/1000.0/60.0;

    alexaString += ("with program name " + programName + " has produced " + String(numberOfParts) + "
pieces with average cycle time of " + String(averageCycleTimeInMin, 2) + " minutes, ");

    String prossedData = "{\"program\":\"" + programName + "\",\"numberOfParts\":\"" +
String(numberOfParts) + "\",\"averageCycleTimeInMin\":\"" + String(averageCycleTimeInMin, 2) +
"\",\"standardDeviationInMin\":\"" + String(SDinMIn, 2) + "\"}";


    // Serial.println("**********************************************************************");
    // Serial.println("Program: " + programName);
    // Serial.println("Number of parts made: " + String(numberOfParts));
    // Serial.println("Average cycle time (min): " + String(averageCycleTimeInMin));
    // Serial.println("Standard deviation (min): " + String(SDinMIn));
    // Serial.println("Generated data:");
    // Serial.println(prossedData);
    // Serial.println("**********************************************************************");
    // Serial.println();

    return prossedData;
}


int getNumberOfValues(String text)
{
    int result = 0;
    for(int i = 0; i < text.length(); i++)
    {
        if(text[i] == '|')
        {
            result++;
```

```
        }
    }
    return result-1;
}
```

A.4 Code on Particle IDE for MQTT implementation on hosted message broker on AWS

```
//  Created by Mahmoud Parto
//  Copyright Â© 2017. All rights reserved.

// This #include statement was automatically added by the Particle IDE.
#include "MQTT.h"

// ************************************************** MQTT
// call back function for incoming data from MQTT
void callback(char* topic, byte* payload, unsigned int length);
// MQTT configuration
MQTT client("mb8.iotfm.org", 1883, callback);
String asset = "VMC-3Axis";

void parseMTConnectDataAndPublishMQTT(String MTConnectString);

void setup()
{
    Serial.begin(250000); // USB Serial
    Time.zone(-5);
    // connect to the MQTT server
    client.connect(asset);
    if (client.isConnected())
    {
        client.subscribe("Asset/" + asset + "/MTConnect");
    }
}

void loop()
{
    // MQTT
    if (client.isConnected())
        client.loop();
}


/*********************************************************************************************
************************************
 * Function Name  : parseMTConnectDataAndPublishMQTT
 * Input          : MTConnectString: MTConnect with standard tags
 * Description    : parse the data and values and publish them with MQTT (each variable and its value)

*********************************************************************************************
************************************/
void parseMTConnectDataAndPublishMQTT(String MTConnectString)
{
    String timeStandard = Time.format(Time.now(), TIME_FORMAT_ISO8601_FULL);

    int index = 0;
```

```
    String data = "";
    String value = "";

    String threeQuoteSplits[3];

    String temp = "";
    int length = MTConnectString.length();
    for(int i = 0 ; i < length; i++)
    {
       i--;
       while(i++ < length & MTConnectString[i] != '"')
       {
          char chr = MTConnectString[i];
          temp += String(chr);
       }
       threeQuoteSplits[2] = threeQuoteSplits[1];
       threeQuoteSplits[1] = threeQuoteSplits[0];
       threeQuoteSplits[0] = temp.trim();
       temp = "";

       String jsonForMQTT = "{\"dateTime\": \"" + timeStandard + "\", \"assetId\": \"" + asset + "\",
\"dataItemId\": \"" +  threeQuoteSplits[2] + "\", \"value\": \"" + threeQuoteSplits[0] + "\"}";

       if (client.isConnected())
          client.publish("Asset/" + asset + "/" + threeQuoteSplits[2], jsonForMQTT);
    }
}


/*********************************************************************************************
************************************
 * Function Name  : callback
 * Description    : receiving message from subscribed MQTTs

*********************************************************************************************
************************************/
void callback(char* topic, byte* payload, unsigned int length)
{
    char payloadChar[length + 1];
    memcpy(payloadChar, payload, length);
    payloadChar[length] = NULL;

    Serial.println("Received topic:" + String(topic));
    Serial.println("Received payload:" + String(payloadChar));

    parseMTConnectDataAndPublishMQTT(payloadChar);
}
```

A.5 Code on Particle IDE for sending text message, email, and data insert to database using developed functions and integration of Twilio.com, Google Scripts, and Google Cloud Platform

```
//  Created by Mahmoud Parto
//  Copyright Â© 2017. All rights reserved.

String body = "I just woke up!!";

const int accelPin = A0;

void setup()
{
   Serial.begin(9600);
   pinMode(accelPin,INPUT);

   // // Send an text message with Twilo to Mahmoud's phone number
   // Particle.publish("textMessageToMahmoud", body, PRIVATE);

   // // Send an email with Google Scripts from GoergiaTech.PMRC to Mahmoud.Parto@gmail.com
   // Particle.publish("sendEmailToMahmoud", body, PRIVATE);

   // Insert 1200 as Spindle Speed in Google Cloud
   Particle.publish("spindleSpeedGoogleCloudFunction", "1200", PRIVATE);
   // To see the data:
   // Go to terminal of Google Cloud
   // type: gcloud beta pubsub subscriptions create test_sub --topic spindle-speed-topic
   // to pull data: gcloud beta pubsub subscriptions pull test_sub --auto-ack --max-messages 100
}

void loop()
{
   float accelInG = abs(analogRead(accelPin) - 2047.5) / 2047.5 * 200.0; // 200G is the max
   Serial.println(accelInG);

   if(accelInG>50)
   {
      String message = "I had an accident with " + String(accelInG) + "G!";
      // Send an text message with Twilo to Mahmoud's phone number
      Particle.publish("textMessageToMahmoud", message, PRIVATE);

      // Send an email with Google Scripts from GoergiaTech.PMRC
      Particle.publish("sendEmailToMahmoud", message, PRIVATE);
   }
}
```

A.6 Compiled code on Particle IDE for SmartBrain

```
//  Created by Mahmoud Parto
//  Copyright Â© 2017. All rights reserved.

// This #include statement was automatically added by the Particle IDE.
#include "MQTT.h"

// This #include statement was automatically added by the Particle IDE.
#include <ArduinoJson.h>

// This #include statement was automatically added by the Particle IDE.
#include "readFromSerial.h"
```

```
// This #include statement was automatically added by the Particle IDE.
#include "spindleCycleAnalysis.h"

// This #include statement was automatically added by the Particle IDE.
#include <SparkJson.h>

// ************************************************ Public Variables
char spindleCycleTimesBuffer[2560];
char MTConnectDataBuffer[2560];
char MTConnectStandardDataBuffer[2560];
char processedDataBuffer[2560];
char alexaDataBuffer[2560];

const int accelPin = A0;
Servo servo;
bool goBackAndForth = false;
int spindleSpeed = 0;

// ************************************************ Threads
Thread* threadMain;
Thread* threadMisc;
Timer timerDB(60000, everyMinute);

// ************************************************ MQTT
// call back function for incoming data from MQTT
void callback(char* topic, byte* payload, unsigned int length);
// MQTT configuration
MQTT client("mb8.iotfm.org", 1883, callback);
String asset = "OkumaAMPF";// "VMC-3Axis";

void parseMTConnectDataAndPublishMQTT(String MTConnectString);

void setup()
{
   Serial.begin(250000); // USB Serial
   Serial.println("Just woke up!");
   Serial1.begin(250000); // TX RX Serial
   Serial1.setTimeout(3000);

   Time.zone(-5);

   // connect to the MQTT server
   client.connect(asset);
   if (client.isConnected())
   {
      client.subscribe("Asset/" + asset + "/MTConnect");
   }

   Particle.variable("spCycles", spindleCycleTimesBuffer);
   // Other photon device id: 290038001247343438323536
   // https://api.particle.io/v1/devices/-/MTCOriginal?access_token=-

   // https://api.particle.io/v1/devices/-/spCycles?access_token=-
   Particle.variable("MTCOriginal", MTConnectDataBuffer);
   // https://api.particle.io/v1/devices/-/MTCOriginal?access_token=-
```

```
    Particle.variable("MTCStandard", MTConnectStandardDataBuffer);
    // https://api.particle.io/v1/devices/-/MTCStandard?access_token=-
    Particle.variable("processed", processedDataBuffer);
    // https://api.particle.io/v1/devices/-/processed?access_token=-
    Particle.variable("programs", alexaDataBuffer);
    // https://api.particle.io/v1/devices/-/programs?access_token=-

    // cloud function e.g. for alexa
    Particle.function("servo", rotateServo);

    // email current status
    Particle.function("emailStatus", emailStatus);

    pinMode(accelPin,INPUT);

    // Insert 1200 as Spindle Speed in Google Cloud
    Particle.publish("spindleSpeedGoogleCloudFunction", "1200", PRIVATE);
    // To see the data:
    // Go to terminal of Google Cloud
    // type: gcloud beta pubsub subscriptions create test_sub --topic spindle-speed-topic
    // to pull data: gcloud beta pubsub subscriptions pull test_sub --auto-ack --max-messages 100

    servo.attach(A5);

    // Start new threads
    threadMain = new Thread("threadMain", threadMainFunc);
    threadMisc = new Thread("threadMisc", threadMiscFunc);

    // start the timer to insert data to database
    // timerDB.start();
}

// ************************************************* Main Thread
// do all of the high frequency tasks here
os_thread_return_t threadMainFunc()
{
    Serial.println("threadMainFunc");
    // Start never ending loop
    while(true)
    {
        // checkAccelerometer();

        // MQTT
        if (client.isConnected())
            client.loop();
    }
}

// ************************************************* Miscellaneous Thread
// do everything here and let main loop take care of high frequency tasks
os_thread_return_t threadMiscFunc()
{
    Serial.println("threadMiscFunc");
    // Start never ending loop
    while(true)
    {
```

```
      String wholeIncomingData = readWholeDataFromSerial();
      String MTConnectData = getDataFromMTConnect(wholeIncomingData);
      String incomingCycles = getCyclesFromMTConnect(wholeIncomingData);

      if (MTConnectData != "")
      {
         strcpy(MTConnectDataBuffer, MTConnectData.c_str());
         String standardMTC = mapMTConnectToStandard(MTConnectData);
         strcpy(MTConnectStandardDataBuffer, standardMTC.c_str());
         // Serial.println("************* MTConnect Data: ***************");
         // Serial.println(MTConnectData);
         // Serial.println();

         // publish on MQTT
         if (client.isConnected())
         {
            String variable = "MTConnect";
            String timeStandard = Time.format(Time.now(), TIME_FORMAT_ISO8601_FULL);
            String jsonForMQTT = "{\"dateTime\": \"" + timeStandard + "\", \"assetId\": \"" + asset + "\",
\"dataItemId\": \"" +  variable + "\", \"value\": \"" + standardMTC + "\"}";
            client.publish("Asset/" + asset + "/" + variable, jsonForMQTT);
            // Particle.publish("Asset/" + asset + "/" + variable, jsonForMQTT);

            // variable = "MTConnectOriginal";
            // jsonForMQTT = "{\"dateTime\": \"" + timeStandard + "\", \"assetId\": \"" + asset + "\",
\"dataItemId\": \"" +  variable + "\", \"value\": \"" + MTConnectData + "\"}";
            // client.publish("Asset/" + asset + "/" + variable, jsonForMQTT);

            // parseMTConnectDataAndPublishMQTT(standardMTC);
         }

         // get the spindleSpeed
         // String spindleSpeedString = getDataWithDataName(standardMTC, "spindleSpeed");
         // spindleSpeed = spindleSpeedString.toInt();
      }
      // spindleSpeed = random(0,2)*3000; // 0 or 3000

      // process the incoming data from spindle cycle times
      if (incomingCycles != "")
      {
         strcpy(spindleCycleTimesBuffer, incomingCycles.c_str());
         String processedDataJSON = processIncomingData(incomingCycles);
         strcpy(processedDataBuffer, processedDataJSON.c_str());

         // Serial.println("************* Spindle Cycles: ***************");
         // Serial.println(incomingCycles);
         // Serial.println("************ processedDataJSON: *************");
         // Serial.println(processedDataJSON);
         // Serial.println("alexaDataBuffer: " + String(alexaDataBuffer));
      }
   }
}


// timer function for inserting data to database
void everyMinute()
```

```
{
   String spindleSpeedString = String(spindleSpeed);
   Particle.publish("MTConnectToMySQL",spindleSpeedString, PRIVATE);
}


void loop()
{
   if(goBackAndForth)
   {
      servo.write(90);
      delay(1500);
      servo.write(45);
      delay(1500);
   }
}


// check the accelerometer and send an email and a message if it was more than 50G
void checkAccelerometer()
{
   float accelInG = abs(analogRead(accelPin) - 2047.5) / 2047.5 * 200.0; // 200G is the max
   Serial.println(accelInG);

   if(accelInG > 70)
   {
      String message = "Shock with " + String(accelInG, 2) + "G at " + Time.timeStr();

      // Send an text message with Twilo to Mahmoud's phone number
      Particle.publish("textMessageToMahmoud", message, PRIVATE);

      // Send an email with Google Scripts from GoergiaTech.PMRC to Mahmoud.Parto@gmail.com
      Particle.publish("sendEmailToMahmoud", message, PRIVATE);

      Particle.publish("alert", message);
   }
}

// rotate the servo to a given position. used to demonstrate the Alexa
int rotateServo(String command)
{
   if(command == "on")
   {
      goBackAndForth = true;
   }
   else if (command == "off")
   {
      goBackAndForth = false;
   }
   else
   {
      int pos = command.toInt();
      servo.write[12];
   }
   return 1;
}
```

```
/*****************************************************************************************
************************************
 * Function Name  : mapMTConnectToStandard
 * Input          : MTConnectString: Json form of original data of MTConnect with dataItemIds
 * Description    : Maps the dataItemIds with dedicated names to standarize the string
 * Output Type    : String

*****************************************************************************************
************************************/
String mapMTConnectToStandard(String MTConnectString)
{
   MTConnectString = MTConnectString.replace("c2", "spindleSpeed");
   MTConnectString = MTConnectString.replace("c3", "spindleOverride");
   MTConnectString = MTConnectString.replace("cl3", "load");
   MTConnectString = MTConnectString.replace("cm", "rotaryMode");
   MTConnectString = MTConnectString.replace("Cloadc", "Normal");
   MTConnectString = MTConnectString.replace("estop", "emergencyStop");
   MTConnectString = MTConnectString.replace("msg", "message");
   MTConnectString = MTConnectString.replace("avail", "availability");
   MTConnectString = MTConnectString.replace("dev_asset_chg", "assetChanged");
   MTConnectString = MTConnectString.replace("dev_asset_rem", "assetRemoved");
   MTConnectString = MTConnectString.replace("Fovr", "feedover");
   MTConnectString = MTConnectString.replace("Frt", "feedrate");
   MTConnectString = MTConnectString.replace("Ppos", "pathPosition");
   MTConnectString = MTConnectString.replace("p2", "powerStatus");
   MTConnectString = MTConnectString.replace("cn2", "block");
   MTConnectString = MTConnectString.replace("cn3", "controllerMode");
   MTConnectString = MTConnectString.replace("cn4", "line");
   MTConnectString = MTConnectString.replace("cn5", "program");
   MTConnectString = MTConnectString.replace("cn6", "execution");
   MTConnectString = MTConnectString.replace("cnt1", "toolId");
   MTConnectString = MTConnectString.replace("x2", "loadX");
   MTConnectString = MTConnectString.replace("x3", "actualX");
   MTConnectString = MTConnectString.replace("x4", "commandedX");
   MTConnectString = MTConnectString.replace("y2", "loadY");
   MTConnectString = MTConnectString.replace("y3", "actualY");
   MTConnectString = MTConnectString.replace("y4", "commandedY");
   MTConnectString = MTConnectString.replace("z2", "loadZ");
   MTConnectString = MTConnectString.replace("z3", "actualZ");
   MTConnectString = MTConnectString.replace("z4", "commandedZ");
   return MTConnectString;
}

/*****************************************************************************************
************************************
 * Function Name  : getDataWithDataName
 * Input          : MTConnectString: MTConnect with standard tags
 * Description    : fins the value of the dataName from the given json
 * Output Type    : String

*****************************************************************************************
************************************/
String getDataWithDataName(String MTConnectString, String dataName)
```

```
{
   String value = "0";
   String data = "";
   int length = MTConnectString.length();
   for(int i = 0 ; i < length; i++)
   {
      if(MTConnectString[i] == ':')
      {
         if(data.endsWith(dataName + "\""))
         {
            i++;
            // Serial.println("It ended with: " + dataName + " Here is the data: " + data);
            char chr;
            do
            {
               chr = MTConnectString[++i];
               if(chr != '\"')
                  value += String(chr);
            }
            while(chr != '\"' || i == length - 1);
            break;
         }
      }
      data += MTConnectString[i];
   }
   return value;
}


/**************************************************************************************************
************************************
 * Function Name  : parseMTConnectDataAndPublishMQTT
 * Input          : MTConnectString: MTConnect with standard tags
 * Description    : parse the data and values and publish them with MQTT (each variable and its value)

**************************************************************************************************
************************************/
void parseMTConnectDataAndPublishMQTT(String MTConnectString)
{
   String timeStandard = Time.format(Time.now(), TIME_FORMAT_ISO8601_FULL);

   int index = 0;
   String data = "";
   String value = "";

   String threeQuoteSplits[3];

   String temp = "";
   int length = MTConnectString.length();
   for(int i = 0 ; i < length; i++)
   {
      i--;
      while(i++ < length & MTConnectString[i] != "")
      {
         char chr = MTConnectString[i];
         temp += String(chr);
```

```
        }
        threeQuoteSplits[2] = threeQuoteSplits[1];
        threeQuoteSplits[1] = threeQuoteSplits[0];
        threeQuoteSplits[0] = temp.trim();
        temp = "";

        String jsonForMQTT = "{\"dateTime\": \"" + timeStandard + "\", \"assetId\": \"" + asset + "\",
\"dataItemId\": \"" +  threeQuoteSplits[2] + "\", \"value\": \"" + threeQuoteSplits[0] + "\"}";

        if (client.isConnected())
            client.publish("Asset/" + asset + "/" + threeQuoteSplits[2], jsonForMQTT);
    }
}

/*********************************************************************************************
************************************
 * Function Name  : callback
 * Description    : receiving message from subscribed MQTTs

*********************************************************************************************
************************************/
void callback(char* topic, byte* payload, unsigned int length)
{
    char payloadChar[length + 1];
    memcpy(payloadChar, payload, length);
    payloadChar[length] = NULL;

    // Serial.println("Received topic:" + String(topic));
    // Serial.println("Received payload:" + String(payloadChar));
}



int emailStatus(String command)
{
    String message = "This is the current status";
    Particle.publish("sendEmailToMahmoud", message, PRIVATE);
    return 1;
}
```

A.7 HTML code for the main webpage of website

```
ï»¿<!DOCTYPE html>
<!-- saved from url=(0028)http://pmrc.marc.gatech.edu/ -->
<html lang="en" dir="ltr" xmlns:content="http://purl.org/rss/1.0/modules/content/"
xmlns:dc="http://purl.org/dc/terms/" xmlns:foaf="http://xmlns.com/foaf/0.1/"
xmlns:og="http://ogp.me/ns#" xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:sioc="http://rdfs.org/sioc/ns#" xmlns:sioct="http://rdfs.org/sioc/types#"
xmlns:skos="http://www.w3.org/2004/02/skos/core#"
xmlns:xsd="http://www.w3.org/2001/XMLSchema#" class="js">
<head profile="http://www.w3.org/1999/xhtml/vocab">
```

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
<link rel="shortcut icon" href="http://pmrc.marc.gatech.edu/sites/all/themes/gt_subtheme/favicon.ico"
type="image/vnd.microsoft.icon">
<link rel="shortlink" href="http://pmrc.marc.gatech.edu/node/1">
<link rel="canonical" href="http://pmrc.marc.gatech.edu/home">
<meta name="Generator" content="Drupal 7 (http://drupal.org)">
<title>Home | PMRC | Georgia Institute of Technology | Atlanta, GA</title>
<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link type="text/css" rel="stylesheet" href="./index_files/font-awesome.min.css" media="all">
<link type="text/css" rel="stylesheet" href="./index_files/css" media="all">
<link type="text/css" rel="stylesheet" href="./index_files/css(1)" media="all">
<link rel="stylesheet" href="css/contentGeorgiaTech.css">
<link rel="stylesheet" href="css/layoutGeorgiaTech.css">



<script async="" src="./index_files/analytics.js.download"></script>
<script type="text/javascript"
src="./index_files/js_UWQINlriydSoeSiGQxToOUdv493zEa7dpsXC1OtYlZU.js.download"></script>
<script type="text/javascript"
src="./index_files/js_v9s9BlH6_53gzKzzUu9FU_COwZfuWBZO1P0bOflB0nc.js.download"></script>
<script type="text/javascript"
src="./index_files/js_I8yX6RYPZb7AtMcDUA3QKDZqVkvEn35ED11_1i7vVpc.js.download"></script>
<script type="text/javascript">
<!--//--><![CDATA[//><!--
(function(i,s,o,g,r,a,m){i["GoogleAnalyticsObject"]=r;i[r]=i[r]||function(){(i[r].q=i[r].q||[]).push(arguments
)},i[r].l=1*new
Date();a=s.createElement(o),m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBe
fore(a,m)})(window,document,"script","//www.google-analytics.com/analytics.js","ga");ga("create", "UA-
61618700-1", {"cookieDomain":"auto"});ga("set", "anonymizeIp", true);ga("send", "pageview");
//--><!]]>
</script>
<script type="text/javascript"
src="./index_files/js_4lYeY6E5zeARk9RwRDf7By5Cy9QqSFa0N5PphC8vE8Y.js.download"></script>
<script type="text/javascript" src="./index_files/js_y0lodT5laN_2kFiSGnUiVkKXDiNh-
Y69x3a71RaUNas.js.download"></script>
<script type="text/javascript">
<!--//--><![CDATA[//><!--
jQuery.extend(Drupal.settings,
{"basePath":"\/","pathPrefix":"","ajaxPageState":{"theme":"gt_subtheme","theme_token":"C5COycts98FP
mI6cFL4GjzMJmW9POHWdtwDjOMwOiig","js":{"sites\/all\/modules\/flexslider\/assets\/js\/flexslider.loa
d.js":1,"misc\/jquery.js":1,"misc\/jquery.once.js":1,"misc\/drupal.js":1,"sites\/all\/modules\/gt_ct_super_blo
ck\/js\/gt_ct_super_block.js":1,"sites\/all\/modules\/google_analytics\/googleanalytics.js":1,"0":1,"sites\/all\
/libraries\/flexslider\/jquery.flexslider-
min.js":1,"sites\/all\/themes\/gt\/js\/html5shiv.js":1,"sites\/all\/themes\/gt\/js\/gt.js":1},"css":{"modules\/syst
em\/system.base.css":1,"modules\/system\/system.menus.css":1,"modules\/system\/system.messages.css":1,
"modules\/system\/system.theme.css":1,"sites\/all\/modules\/date\/date_api\/date.css":1,"modules\/field\/the
me\/field.css":1,"sites\/all\/modules\/gt_ct_carousel_slider\/styles\/css\/gt_ct_carousel_slider.css":1,"sites\/a
ll\/modules\/gt_ct_super_block\/styles\/css\/gt_ct_super_block.css":1,"modules\/node\/node.css":1,"module
s\/search\/search.css":1,"modules\/user\/user.css":1,"sites\/all\/modules\/views\/css\/views.css":1,"sites\/all\/
modules\/ckeditor\/css\/ckeditor.css":1,"sites\/all\/modules\/jwfilter\/jwfilter.css":1,"sites\/all\/modules\/cto
ols\/css\/ctools.css":1,"sites\/all\/modules\/flexslider\/assets\/css\/flexslider_img.css":1,"sites\/all\/libraries\/
flexslider\/flexslider.css":1,"https:\/\/maxcdn.bootstrapcdn.com\/font-awesome\/4.3.0\/css\/font-
awesome.min.css":1,"https:\/\/fonts.googleapis.com\/css?family=Roboto:400,400italic,500,500italic,700,70
```

0italic,300":1,"https:\/\/fonts.googleapis.com\/css?family=Roboto+Slab:400,700":1,"sites\/all\/themes\/gt\/css\/reset.css":1,"sites\/all\/themes\/gt\/css\/default.css":1,"sites\/all\/themes\/gt\/css\/fonts.css":1,"sites\/all\/themes\/gt\/css\/typography.css":1,"sites\/all\/themes\/gt\/css\/layout.css":1,"sites\/all\/themes\/gt\/css\/blocks.css":1,"sites\/all\/themes\/gt\/css\/content.css":1,"sites\/all\/themes\/gt\/css\/editor.css":1,"sites\/all\/themes\/gt\/css\/print.css":1,"sites\/all\/themes\/gt_subtheme\/css\/gt_subtheme.css":1,"sites\/all\/themes\/gt_subtheme\/css\/custom.css":1}},"googleanalytics":{"trackOutbound":1,"trackMailto":1,"trackDownload":1,"trackDownloadExtensions":"7z|aac|arc|arj|asf|asx|avi|bin|csv|doc(x|m)?|dot(x|m)?|exe|flv|gif|gz|gzip|hqx|jar|jpe?g|js|mp(2|3|4|e?g)|mov(ie)?|msi|msp|pdf|phps|png|ppt(x|m)?|pot(x|m)?|pps(x|m)?|ppam|sld(x|m)?|thmx|qtm?|ra(m|r)?|sea|sit|tar|tgz|torrent|txt|wav|wma|wmv|wpd|xls(x|m|b)?|xlt(x|m)|xlam|xml|z|zip"},"urlIsAjaxTrusted":{"\/search\/node":true},"flexslider":{"optionsets":{"default":{"namespace":"flex-","selector":".slides
\u003E
li","easing":"swing","direction":"horizontal","reverse":false,"smoothHeight":false,"startAt":0,"animationSpeed":500,"initDelay":0,"useCSS":true,"touch":true,"video":false,"keyboard":true,"multipleKeyboard":false,"mousewheel":0,"controlsContainer":".flex-control-nav-
container","sync":"","asNavFor":"","itemWidth":0,"itemMargin":0,"minItems":0,"maxItems":0,"move":0,"animation":"fade","slideshow":true,"slideshowSpeed":"4000","directionNav":true,"controlNav":true,"prevText":"Previous","nextText":"Next","pausePlay":false,"pauseText":"Pause","playText":"Play","randomize":false,"thumbCaptions":false,"thumbCaptionsBoth":false,"animationLoop":true,"pauseOnAction":true,"pauseOnHover":false,"manualControls":""}},"instances":{"carousel-12":"default"}}});
//--><!]]>
</script>
<link rel="stylesheet" href="./index_files/base.css">
<!--[if lte IE 8]>
<style type="text/css" media="all">@import "/sites/all/themes/gt/css/ie.css";</style>
<![endif]-->

<!-- Mahmoud Parto -->
<link rel="stylesheet" href="css/bootstrap.min.css">
<script src="js/popper.min.js"></script>
<script src="js/bootstrap.min.js"></script>

<script src="js/createTableFromJsonMahmoudParto.js"></script>

<script src="js/highcharts.js"></script>
<script src="js/highcharts-more.js"></script>
<script src="js/exporting.js"></script>
<script src="js/createGraphMahmoudParto.js"></script>

<script src="js/jquery.min.js"></script>
<script src="js/jquery-ui.js"></script>
<script src="js/notify.js"></script>
<!-- Mahmoud Parto JavaScript to get data from SmartBrain -->
<script src="js/main.js"></script>

</head>
<body class="html front not-logged-in no-sidebars page-node page-node- page-node-1 node-type-horizontal-landing-page user-role-anonymous front-page-title-hidden">
<!--[if IE 7]> <![endif]-->
<!--[if IE 8]> <![endif]-->
<!--[if IE 9]> <![endif]-->

Skip to content

```
<div id="page">
<header id="masthead">
<section id="identity">
<div id="identity-wrapper" class="clearfix">
<h1 id="gt-logo">
<img alt="PMRC | Georgia Institute of Technology | Atlanta, GA" class="uploaded-logo-file" src="./index_files/precisionmachiningresearchconsortium-solid-2lines-white_0.png">
<a href="http://www.gatech.edu/" id="gt-logo-mothership-link" title="Georgia Institute of Technology">Georgia Institute of Technology</a>
<a href="http://manufacturing.gatech.edu/" id="gt-logo-secondary-link" title="PMRC | Georgia Institute of Technology | Atlanta, GA">PMRC | Georgia Institute of Technology | Atlanta, GA</a>
</h1>
</div>
</section><!-- /#identity -->
<section id="primary-menus">
<div id="primary-menus-wrapper" class="clearfix">
<a id="primary-menus-toggle" class="hide-for-desktop"><span>Menu</span></a>
<div id="primary-menus-off-canvas" class="off-canvas">
<a id="primary-menus-close" class="hide-for-desktop"><span>Close</span></a>
<nav>
<div id="main-menu-wrapper">
<ul class="menu">
<li class="first leaf main-menu-link-310"><a href="http://pmrc.marc.gatech.edu/home" class="active"><span>Welcome</span></a></li>
<li class="leaf main-menu-link-952"><a href="http://pmrc.marc.gatech.edu/about-us"><span>About Us</span></a></li>
<li class="expanded main-menu-link-737">
<a href="http://pmrc.marc.gatech.edu/faculty-and-staff"><span>Faculty</span></a><ul class="menu">
<li class="first leaf main-menu-link-810"><a href="http://pmrc.marc.gatech.edu/dr-thomas-kurfess"><span>Dr. Thomas Kurfess</span></a></li>
<li class="leaf main-menu-link-656"><a href="http://pmrc.marc.gatech.edu/dr-steven-liang"><span>Dr. Steven Y. Liang</span></a></li>
<li class="leaf main-menu-link-744"><a href="http://pmrc.marc.gatech.edu/dr-j-rhett-mayor"><span>Dr. J. Rhett Mayor</span></a></li>
<li class="leaf main-menu-link-666"><a href="http://pmrc.marc.gatech.edu/dr-shreyes-melkote"><span>Dr. Shreyes N. Melkote</span></a></li>
<li class="last leaf main-menu-link-812"><a href="http://pmrc.marc.gatech.edu/dr-christopher-saldana"><span>Dr. Christopher J. Saldana</span></a></li>
</ul>
</li>
<li class="leaf main-menu-link-1688"><a href="http://pmrc.marc.gatech.edu/complete-pmrc-roster"><span>Members</span></a></li>
<li class="leaf main-menu-link-568"><a href="http://pmrc.marc.gatech.edu/list-equipment"><span>Equipment</span></a></li>
<li class="leaf main-menu-link-950"><a href="http://pmrc.marc.gatech.edu/sponsors-partnerships"><span>Partnerships</span></a></li>
<li class="last leaf main-menu-link-651"><a href="http://pmrc.marc.gatech.edu/contact-information"><span>Contact</span></a></li>
</ul>
</div>
<div id="action-items-wrapper">
</div>
</nav>
```

```
                <div id="utility">
                  <div class="row clearfix">

                    <!-- #utility-links -->

                    <nav id="utility-links">
                      <ul class="menu">
                        <li class="mothership ulink"><a href="http://www.gatech.edu/">Georgia Tech
Home</a></li>
                        <li class="campus-map ulink"><a href="http://map.gtalumni.org/">Map</a></li>
                        <li class="directories ulink"><a
href="http://www.directory.gatech.edu/">Directory</a></li>
                        <li class="offices ulink"><a href="http://www.gatech.edu/offices-and-
departments">Offices</a></li>
                      </ul>
                    </nav>          <!-- /#utility-links -->

                    <div id="social-media-links-wrapper">
                    </div>
                  </div>
                </div><!-- /#utility -->
              </div>
              <div id="site-search" class="search-gt">
                <a href="http://search.gatech.edu/" id="site-search-container-switch">Search</a>
                <div id="site-search-container">
                  <div id="search-gt"><form action="http://search.gatech.edu/search"
method="get"><input value="" name="q" id="q" class="form-text" type="text"><input name="site"
value="default_collection" type="hidden"><input name="client" value="default_frontend"
type="hidden"><input name="output" value="xml_no_dtd" type="hidden"><input
name="proxystylesheet" value="default_frontend" type="hidden"></form></div>
                </div>
              </div>
            </div><!-- /#primary-menus-wrapper -->
            <div id="breadcrumb" class="hide-for-mobile">
              <div class="row clearfix">
                <ul><li class="element-invisible">You are here: </li><li class="breadcrumb-item first"><a
href="http://www.gatech.edu/">GT Home</a></li></ul>
              </div>
            </div><!-- /#breadcrumb -->
          </section><!-- /#primary-menus -->


      </header><!-- /#masthead -->
      <section id="main">
        <div class="row clearfix">
          <div id="page-title">
            <h2 class="title">Home</h2>
          </div>


          <div id="content-lead">
            <div class="region region-content-lead block-count-1 clearfix">
              <div id="block-nodeblock-12" class="block block-nodeblock full-width  block-region-
weight-1 odd">

                <div class="content block-body clearfix">
```

```
                    <div id="node-12" class="node node-gt-ct-carousel" about="/carousel/12"
typeof="sioc:Item foaf:Document">
                        <div class="content node-body clearfix">

                        </div>

                    </div>
                </div>

            </div>
        </div>
    </div>

    <div class="no-sidebars" id="content">

        <div class="region region-content block-count-1 clearfix">
            <div id="block-system-main" class="block block-system   block-region-weight-1 odd">

                <div class="content block-body clearfix">
                    <div id="node-1" class="node node-horizontal-landing-page node-full clearfix"
about="/home" typeof="sioc:Item foaf:Document">

                        <span property="dc:title" content="Home" class="rdf-meta element-
hidden"></span>

                        <div class="content">

                            <div class="body-row">
                                <div class="row-content clearfix">
                                    <h3>Welcome to the Precision Machining Research Consortium</h3>
                                    <p>The Precision Machining Research Consortium at Georgia Tech has been
one of the pioneering laboratories in the world for developing current state of the art solutions in
manufacturing. The PMRC links together the&amp;nbsp;knowledge of leading experts in the field for a
combined total of&amp;nbsp;300+ publications, 80+ years of faculty experience, and&amp;nbsp;20+
federal/industry partnerships to&amp;nbsp;design, improve, and optimize the machining process.</p>
                                </div>
                            </div>




                            <div class="container">
                                <div style="text-align:center;">
                                    <img src="img/VCU400A-800x600.png" style="vertical-align:middle; max-
width: 100%;"/>
                                    <p>Picture from www.mazakindia.in</p>
                                    <h2>VMC-3Axis</h2>
                                    <br /><br />
                                </div>
                                <span id="alert" style="color:green"></span>

                                <div style="height:40px"></div>
                                <div id="containerSpindleSpeedHallEffect" style="min-width: 310px; max-
width: 400px; height: 300px; margin: 0 auto"></div>
```

```
                        <div id="containerSpindleSpeedMTConnect" style="min-width: 310px; max-
width: 400px; height: 300px; margin: 0 auto"></div>



                        <script src="js/containerSpindleSpeedHallEffect.js"></script>

                        <script src="js/containerSpindleSpeedMTConnect.js"></script>

                        <div style="height:100px"></div>
                        <div id="containerNumberOfParts" style="min-width: 310px; height: 400px;
margin: 0 auto"></div>
                        <div id="containerAverageCycleTimes" style="min-width: 310px; height:
400px; margin: 0 auto"></div>
                        <div id="table1" style="overflow-x: scroll"></div>

                        <div id="containerPast60Minutes" style="min-width: 310px; height: 400px;
margin: 0 auto"></div>
                        <br><br><br><br>
                        <div id="containerPast24Hours" style="min-width: 310px; height: 400px;
margin: 0 auto"></div>
                        <br><br><br><br>
                        <div id="containerPast30Days" style="min-width: 310px; height: 400px;
margin: 0 auto"></div>
                        <br><br><br><br>
                        <div id="containerPast12Months" style="min-width: 310px; height: 400px;
margin: 0 auto"></div>
                        <div id="table2" style="overflow-x: scroll"></div>
                        <div id="table3" style="overflow-x: scroll"></div>

                        <script>
                            listenToAlerts();
                            // start EventListener to listen to alerts coming from Smart Brain
                            function listenToAlerts() {
                                var deviceID = "-";
                                var accessToken = "-";
                                document.getElementById("alert").innerHTML = "Listening for alerts...";
                                var eventSource = new EventSource("https://api.spark.io/v1/devices/" +
deviceID + "/events/?access_token=" + accessToken);

                                eventSource.addEventListener('open', function (e) {
                                    console.log("Opened!");
                                }, false);

                                eventSource.addEventListener('error', function (e) {
                                    console.log("Errored!");
                                }, false);

                                eventSource.addEventListener('alert', function (e) {

                                    var rawData = JSON.parse(e.data);
                                    var tempSpan = document.getElementById("alert");
                                    tempSpan.innerHTML = "Alert message: " + rawData.data + " <br>\ " +
"Machine name: VMC-3Axis" + " <br>\ " + "Core: " + rawData.coreid ;
                                    tempSpan.style.fontSize = "28px";
                                    tempSpan.style.color = "red";
```

```
                    $.notify("Alert: " + rawData.data, "error");

                }, false);
            }
        </script>

    </div>




    </div>


        </div>
    </div>

        </div>
    </div>
</div><!-- /#content -->



    </div>
</section><!-- /#main -->
<!-- #superfooter/ -->
<div class="superfooter-trigger-wrapper clearfix"><a class="js__superfooter-trigger collapsed"
href="http://pmrc.marc.gatech.edu/#superfooter" id="superfooter-trigger">Resources</a></div><section
id="superfooter" class="superfooter-gt-default-full collapsible">
    <div class="row clearfix">
        <div class="superfooter-resource-links" id="gt-default-resource-links">
            <h4 class="title">Georgia Tech Resources</h4>
            <ul class="menu" id="gt-default-resources">
                <li><a href="http://www.gatech.edu/offices-and-departments">Offices &amp;amp;
Departments</a></li>
                <li><a href="http://www.news.gatech.edu/">News Center</a></li>
                <li><a href="http://www.gatech.edu/calendar">Campus Calendar</a></li>
                <li><a href="http://www.specialevents.gatech.edu/">Special Events</a></li>
                <li><a href="http://www.greenbuzz.gatech.edu/">GreenBuzz</a></li>
                <li><a href="http://www.comm.gatech.edu/">Institute Communications</a></li>
                <li><span class="nolink">Visitor Resources</span></li>
                <li class="gt-default-mini-left"><a href="http://www.admission.gatech.edu/visit">Campus
Visits</a></li>
                <li class="gt-default-mini-right"><a
href="http://www.admission.gatech.edu/visit/directions-and-parking">Directions to Campus</a></li>
                <li class="gt-default-mini-left"><a
href="http://www.pts.gatech.edu/visitors/Pages/default.aspx">Visitor Parking Information</a></li>
                <li class="gt-default-mini-right"><a
href="http://www.lawn.gatech.edu/help/GTvisitor.html">GTvisitor Wireless Network
Information</a></li>
                <li class="gt-default-mini-left"><a href="https://pe.gatech.edu/global-learning-
center/">Georgia Tech Global Learning Center</a></li>
                <li class="gt-default-mini-right"><a href="http://www.gatechhotel.com/">Georgia Tech
Hotel &amp;amp; Conference Center</a></li>
```

```
                <li class="gt-default-mini-left"><a href="http://www.gatech.bncollege.com/">Barnes
&amp;amp; Noble at Georgia Tech</a></li>
                <li class="gt-default-mini-right"><a href="http://www.ferstcenter.gatech.edu/">Ferst Center
for the Arts</a></li>
                <li class="gt-default-mini-left"><a href="http://www.ipst.gatech.edu/amp">Robert C.
Williams Paper Museum</a></li>
              </ul>
            </div>
            <div id="gt-footer-links-1" class="superfooter-resource-links">
              <h4 class="title">Colleges, Instructional Sites &amp;amp; Research</h4>
              <ul class="menu">
                <li><span class="nolink">Colleges</span></li>
                <li><a href="http://www.coa.gatech.edu/">College of Architecture</a></li>
                <li><a href="http://www.cc.gatech.edu/">College of Computing</a></li>
                <li><a href="http://www.coe.gatech.edu/">College of Engineering</a></li>
                <li><a href="http://www.cos.gatech.edu/">College of Sciences</a></li>
                <li><a href="http://www.iac.gatech.edu/">Ivan Allen College of Liberal Arts</a></li>
                <li><a href="http://www.scheller.gatech.edu/">Scheller College of Business</a></li>
                <li><span class="nolink">Instructional Sites</span></li>
                <li><a href="http://lorraine.gatech.edu/">Georgia Tech-Lorraine</a></li>
                <li><a href="http://www.shenzhen.gatech.edu/">Georgia Tech-Shenzhen</a></li>
                <li><span class="nolink">Global Footprint</span></li>
                <li><a href="http://www.global.gatech.edu/">Global Engagement</a></li>
                <li><span class="nolink">Research</span></li>
                <li><a href="http://www.research.gatech.edu/">Research at Georgia Tech</a></li>
                <li><a href="http://www.research.gatech.edu/evpr">Executive Vice President for
Research</a></li>
              </ul>
            </div>
            <div id="gt-footer-links-2" class="superfooter-resource-links">
              <h4 class="title">Student &amp;amp; Parent Resources</h4>
              <ul class="menu">
                <li><span class="nolink">Student Resources</span></li>
                <li><a href="http://www.gatech.edu/admissions">Apply</a></li>
                <li><a href="http://www.buzzport.gatech.edu/">BuzzPort</a></li>
                <li><a href="http://www.buzzcard.gatech.edu/">Buzzcard</a></li>
                <li><a href="http://career.gatech.edu/">Career Services</a></li>
                <li><a href="http://profpractice.gatech.edu/">Co-ops &amp;amp; Internships</a></li>
                <li><a href="http://www.commencement.gatech.edu/">Commencement</a></li>
                <li><a href="http://www.library.gatech.edu/">Library</a></li>
                <li><a href="http://studentaffairs.gatech.edu/">Student Affairs</a></li>
                <li><a href="http://startup.gatech.edu/">Student Entrepreneurship</a></li>
                <li><a href="http://oie.gatech.edu/study-abroad">Study Abroad</a></li>
                <li><a href="http://www.tsquare.gatech.edu/">T-Square</a></li>
                <li><span class="nolink">Parent Resources</span></li>
                <li><a href="http://www.parents.gatech.edu/">Parents Program</a></li>
                <li><a href="http://www.deanofstudents.gatech.edu/">Dean of Students</a></li>
                <li><a href="http://www.finaid.gatech.edu/">Scholarships &amp;amp; Financial Aid</a></li>
              </ul>
            </div>
            <div id="gt-footer-links-3" class="superfooter-resource-links">
              <h4 class="title">Employee, Alumni, &amp;amp; Other Resources</h4>
              <ul class="menu">
                <li><span class="nolink">Employees</span></li>
                <li><a href="http://www.af.gatech.edu/">Administration and Finance</a></li>
                <li><a href="http://advising.gatech.edu/">Advising &amp;amp; Teaching</a></li>
```

```
                <li><a href="http://facultygovernance.gatech.edu/">Faculty Governance</a></li>
                <li><a href="http://careers.gatech.edu/">Faculty Hiring</a></li>
                <li><a href="http://www.ohr.gatech.edu/">Human Resources</a></li>
                <li><a href="http://www.provost.gatech.edu/">Office of the Provost</a></li>
                <li><a href="http://www.techworks.gatech.edu/">TechWorks</a></li>
                <li><span class="nolink">Alumni</span></li>
                <li><a href="http://www.gtalumni.org/">Alumni Association</a></li>
                <li><a href="http://gtalumni.org/pages/career">Alumni Career Services</a></li>
                <li><a href="http://gtalumni.org/pages/rollcall">Giving Back to Tech</a></li>
                <li><span class="nolink">Outreach</span></li>
                <li><a href="http://www.atdc.org/">Startup Companies</a></li>
                <li><a href="http://www.innovate.gatech.edu/">Economic Development</a></li>
                <li><a href="http://www.industry.gatech.edu/">Industry Engagement</a></li>
                <li><a href="http://www.gov.gatech.edu/">Government &amp;amp; Community
Partners</a></li>
                <li><a href="http://gtri.gatech.edu/">Georgia Tech Research Institute</a></li>
                <li><a href="http://www.gtpe.gatech.edu/">Professional Education</a></li>
              </ul>
            </div><div id="street-address-info"><a href="http://map.gtalumni.org/"><img alt="Map of
Georgia Tech" src="./index_files/gt-map-image-default.jpg"></a><div class="street-address"><p>Georgia
Institute of Technology<br>North Avenue, Atlanta, GA 30332<br>Phone: 404-894-2000</p></div></div>
          </div>
        </section>  <!-- /#superfooter -->
        <!-- #superfooter/ -->
        <footer id="footer">
          <div class="row clearfix">
            <div id="footer-utility-links">
              <div class="gt-footer-utility-links-wrapper login-link-included"></div><div class="gt-footer-
legal-links-wrapper"><ul class="menu gt-footer-legal-links login-link-included"><li class="first"><a
href="http://www.gatech.edu/emergency/">Emergency Information</a></li><li><a
href="http://www.gatech.edu/legal/">Legal &amp;amp; Privacy Information</a></li><li><a
href="http://www.gatech.edu/accessibility/">Accessibility</a></li><li><a
href="http://www.gatech.edu/accountability/">Accountability</a></li><li><a
href="https://www.gatech.edu/accreditation/">Accreditation</a></li><li class="last"><a
href="http://www.careers.gatech.edu/">Employment</a></li></ul></div><div class="gt-footer-login-
links-wrapper"><ul class="menu gt-footer-login-links"><li class="gt-site-login first"><a
href="http://pmrc.marc.gatech.edu/cas">GT Login</a></li><li class="site-login last"><a
href="http://pmrc.marc.gatech.edu/user">Standard Login</a></li></ul></div>
            </div><div id="footer-logo"><a href="http://www.gatech.edu/"><img alt="Georgia Tech"
src="./index_files/gt-logo-footer.png"></a><p>Â© 2017 Georgia Institute of Technology</p></div>
          </div>
        </footer>  <!-- /#superfooter -->
      </div><!-- /#page -->
      <script type="text/javascript" src="./index_files/js_5idECjjAo-X5YdkT65CaIiodkWmZlZv-
WjSkHlWhoYk.js.download"></script>
      <!--[if IE 7]></div><![endif]-->
      <!--[if IE 8]></div><![endif]-->
      <!--[if IE 9]></div><![endif]-->

<div style='text-align: right;position: fixed;z-index:9999999;bottom: 0; width: 100%;cursor: pointer;line-
height: 0;'><a title="Hosted on free web hosting 000webhost.com. Host your own website for FREE."
target="_blank"
href="https://www.000webhost.com/?utm_source=000webhostapp&utm_campaign=000_logo&utm_medi
um=website_gatechpmrc&utm_content=footer_img"><img
src="https://cloud.githubusercontent.com/assets/23024110/20663010/9968df22-b55e-11e6-941d-
edbc894c2b78.png"  alt="www.000webhost.com"></a></div></body>
```

```
</html>
```

A.8 JavaScript code to generate html table from JSON array

```
function createTableFromJson(json, tableNameContainer) {
    // EXTRACT VALUE FOR HTML HEADER.
    var col = [];
    for (var i = 0; i < json.length; i++) {
        for (var key in json[i]) {
            if (col.indexOf(key) === -1) {
                col.push(key);
            }
        }
    }

    // CREATE DYNAMIC TABLE.
    var table = document.createElement("table");
    table.className = "table table-striped";
    table.style.overflowX = "scroll";
    table.style.verticalAlign = "middle";
    // CREATE HTML TABLE HEADER ROW USING THE EXTRACTED HEADERS ABOVE.

    var tr = table.insertRow(-1);                   // TABLE ROW.
    tr.style.minWidth = "200px;"
    for (var i = 0; i < col.length; i++) {
        var th = document.createElement("th");      // TABLE HEADER.
        th.style.minWidth = "200px;"
        th.innerHTML = col[i];
        tr.appendChild(th);
    }

    // ADD JSON DATA TO THE TABLE AS ROWS.
    for (var i = 0; i < json.length; i++) {

        tr = table.insertRow(-1);
        tr.style.minWidth = "200px;"
        for (var j = 0; j < col.length; j++) {
            var tabCell = tr.insertCell(-1);
            tabCell.innerHTML = json[i][col[j]];
        }
    }

    // FINALLY ADD THE NEWLY CREATED TABLE WITH JSON DATA TO A CONTAINER.
    var divContainer = document.getElementById(tableNameContainer);
    divContainer.innerHTML = "";
    divContainer.appendChild(table);
}
```

A.9 Code on Google Scripts for spindle utilization analytics – main

```
// Created by Mahmoud Parto
// Copyright @2016. All right reserved.

// The webapp would send HTTP request, get XML data, find the spindle speed, and publish it to the
spreadsheet
function getSpindleSpeedFromMTConnectURL(rowMax)
{
  // Prevent the sheet from having more than maximum allowed rows
  if(sheet.getLastRow() > rowMax)
  {
    // shift them up (column A,B,C,D)
    sheet.getRange(3,1,sheet.getLastRow(),4).setValues(sheet.getRange(4,1,sheet.getLastRow(),4).getValue
s());
    // delete the last row
    sheet.deleteRow(rowMax+1);
  }

  // Get the current time
  var now = Utilities.formatDate(new Date(), "EST", "yyyy-MM-dd'T'hh:mm:ss a'Z'");

  // Make an string array for the row
  var rowData = [];
  rowData[0] = now;
  rowData[1] = getSpindleSpeedFromMTCBrain();
  rowData[2] = "0";
  rowData[3] = "0";
  if (rowData[1] != 0)
  {
    rowData[2] = "100";
  }
  rowData[3] = rowData[2];


  // Insert the array to the second row
  var newRange = sheet.getRange(2, 1, 1, rowData.length);
  newRange.setValues([rowData]);

  // Find the last row, and make a new row
  var newRow = sheet.getLastRow() + 1;

  // Insert the array to the new row
  newRange = sheet.getRange(newRow, 1, 1, rowData.length);
  newRange.setValues([rowData]);
}

// Helper function for getMTConnectData to get spindle speed from the MTCBrain
function getSpindleSpeedFromMTCBrain()
{
  var url =
'https://api.particle.io/v1/devices/2f003000114735313638363l/MTCStandard?access_token=1337a25b125
d45c5ff0e73d22960a5021fb71521';
  var response = UrlFetchApp.fetch(url, {'muteHttpExceptions': true});
  var json = response.getContentText();
  var parsedData = JSON.parse(json);
  var resultJSON = parsedData.result;
  //Logger.log("resultJSON: " + resultJSON);
```

```
  var resultJSONParsed = JSON.parse(resultJSON);
  var value = resultJSONParsed.spindleSpeed;
  if(value == undefined)
  {
    value = "0";
  }
  //Logger.log(value);

  return value;
}
```

A.10 Code on Google Scripts for spindle utilization analytics – acquiring spindle speed

```
// Created by Mahmoud Parto
// Copyright @2016. All right reserved.

// The webapp would send HTTP request, get XML data, find the spindle speed, and publish it to the
spreadsheet
function getSpindleSpeedFromMTConnectURL(rowMax)
{
  // Prevent the sheet from having more than maximum allowed rows
  if(sheet.getLastRow() > rowMax)
  {
    // shift them up (column A,B,C,D)
    sheet.getRange(3,1,sheet.getLastRow(),4).setValues(sheet.getRange(4,1,sheet.getLastRow(),4).getValue
s());
    // delete the last row
    sheet.deleteRow(rowMax+1);
  }

  // Get the current time
  var now = Utilities.formatDate(new Date(), "EST", "yyyy-MM-dd'T'hh:mm:ss a'Z'");

  // Make an string array for the row
  var rowData = [];
  rowData[0] = now;
  rowData[1] = getSpindleSpeedFromMTCBrain();
  rowData[2] = "0";
  rowData[3] = "0";
  if (rowData[1] != 0)
  {
    rowData[2] = "100";
  }
  rowData[3] = rowData[2];


  // Insert the array to the second row
  var newRange = sheet.getRange(2, 1, 1, rowData.length);
  newRange.setValues([rowData]);

  // Find the last row, and make a new row
  var newRow = sheet.getLastRow() + 1;

  // Insert the array to the new row
  newRange = sheet.getRange(newRow, 1, 1, rowData.length);
```

```
  newRange.setValues([rowData]);
}

// Helper function for getMTConnectData to get spindle speed from the MTCBrain
function getSpindleSpeedFromMTCBrain()
{
  var url =
'https://api.particle.io/v1/devices/2f003000114735313638363 1/MTCStandard?access_token=1337a25b125
d45c5ff0e73d22960a5021fb71521';
  var response = UrlFetchApp.fetch(url, {'muteHttpExceptions': true});
  var json = response.getContentText();
  var parsedData = JSON.parse(json);
  var resultJSON = parsedData.result;
  //Logger.log("resultJSON: " + resultJSON);
  var resultJSONParsed = JSON.parse(resultJSON);
  var value = resultJSONParsed.spindleSpeed;
  if(value == undefined)
  {
    value = "0";
  }
  //Logger.log(value);

  return value;
}
```

A.11 Code on Google Scripts for spindle utilization analytics – HTTP response functions

```
// Created by Mahmoud Parto
// Copyright @2016. All right reserved.

// Response to received HTTP Get requests
function readHTTPGet(e)
{
  //https://script.google.com/macros/s/AKfycbySCSsALcJIoS6nw8RvX3Mp0brrpMHsbBcTCT9uf3CrqhvC619c/exec?readSpindleSpeedUtilizationEveryMinute
  var readFirstRow = e.parameter.readFirstRow;
  var readTimeStamp = e.parameter.readTimeStamp;
  var readSpindleStatus = e.parameter.readSpindleStatus;
  var readSpindleSpeedUtilizationEveryMinute = e.parameter.readSpindleSpeedUtilizationEveryMinute;
  var readSpindleSpeedUtilizationEveryHour = e.parameter.readSpindleSpeedUtilizationEveryHour;
  var readSpindleSpeedUtilizationEveryDay = e.parameter.readSpindleSpeedUtilizationEveryDay;
  var readSpindleSpeedUtilizationEveryMonth = e.parameter.readSpindleSpeedUtilizationEveryMonth;
  var readAll = e.parameter.readAll;
  var readCell = e.parameter.readCell;


  if (readFirstRow != undefined)
  {
    return sheet.getDataRange().getValues()[1];
  }
  else if (readTimeStamp != undefined)
  {
    return sheet.getRange("A3:A").getValues();
  }
  else if (readSpindleStatus != undefined)
```

```
 {
  return sheet.getRange("D3:D").getValues();
 }
 else if (readSpindleSpeedUtilizationEveryMinute != undefined)
 {
  return sheet.getRange("F2:F").getValues();
 }
 else if (readSpindleSpeedUtilizationEveryHour != undefined)
 {
  return sheet.getRange("H2:H").getValues();
 }
 else if (readSpindleSpeedUtilizationEveryDay != undefined)
 {
  return sheet.getRange("J2:J").getValues();
 }
 else if (readSpindleSpeedUtilizationEveryMonth != undefined)
 {
  return sheet.getRange("L2:L").getValues();
 }
 else if (readAll != undefined)
 {
  return sheet.getDataRange().getValues();
 }
 else if (readCell != undefined)
 {
  var i = readCell.split(",")[0];
  var j = readCell.split(",")[1];
  return sheet.getRange(i, j, 1, 1).getValue();
 }

}
```

A.12 Code on Google Scripts for spindle utilization analytics – analytics

```
// Created by Mahmoud Parto
// Copyright @2016. All right reserved.

// Machine Utilization
// Every minute, calculate utilization percentage and fill out the specified columns
function everyMinute()
{
 // calculate the average of column C and add it to column E = F, then clean the column C
 var CColumn = sheet.getRange("C3:C").getValues().filter(String); //only use the ones with values in
calculation
 var average = 0;
 for (var i=0; i < CColumn.length; i++)
 {
  average += Number(CColumn[i]);
 }
 if(CColumn.length != 0)
 {
  average = average / CColumn.length;
 }
 // find the last element in E
 var EValues = sheet.getRange("E1:E").getValues();
```

```
  var ELastElement = EValues.filter(String).length;
  sheet.getRange(ELastElement + 1,5,1,1).setValue(average); //E
  sheet.getRange(ELastElement + 1,6,1,1).setValue(average); //F

  // Clean C
  sheet.getRange("C2:C").clear();
}

// Every hour
function everyHour()
{
  // calculate the average of column E and add it to column G = H, then clean the column E
  var average = 0;
  for(var i = 2; i <= 60 + 1; i++)
  {
    // check if the cell value is a number
    var cell = sheet.getRange(i, 5, 1, 1).getValue();
    if(!isNaN(parseFloat(cell)) && isFinite(cell))
    {
      average += Number(cell);
    }
  }
  average = average/ 60;
  // find the last element in G
  var GValues = sheet.getRange("G1:G").getValues();
  var GLastElement = GValues.filter(String).length;
  sheet.getRange(GLastElement + 1,7,1,1).setValue(average); //G
  sheet.getRange(GLastElement + 1,8,1,1).setValue(average); //H

  // Clean E
  sheet.getRange("E2:E").clear();
}


// Every day
function everyDay()
{
  // calculate the average of column G and add it to column I = J, then clean the column G
  var average = 0;
  for(var i = 2; i <= 24 + 1; i++)
  {
    // check if the cell value is a number
    var cell = sheet.getRange(i, 7, 1, 1).getValue();
    if(!isNaN(parseFloat(cell)) && isFinite(cell))
    {
      average += Number(cell);
    }
  }
  average = average/ 24;
  // find the last element in I
  var IValues = sheet.getRange("I1:I").getValues();
  var ILastElement = IValues.filter(String).length;
  sheet.getRange(ILastElement + 1,9,1,1).setValue(average); //I
  sheet.getRange(ILastElement + 1,10,1,1).setValue(average); //J

  // Clean G
```

```
  sheet.getRange("G2:G").clear();
}

// Every month
function everyMonth()
{
  // calculate the average of column I and add it to column K = L, then clean the column I
  var average = 0;
  for(var i = 2; i <= 30 + 1; i++)
  {
    // check if the cell value is a number
    var cell = sheet.getRange(i, 9, 1, 1).getValue();
    if(!isNaN(parseFloat(cell)) && isFinite(cell))
    {
      average += Number(cell);
    }
  }
  average = average/ 30;
  // find the last element in K
  var KValues = sheet.getRange("K1:K").getValues();
  var KLastElement = KValues.filter(String).length;
  sheet.getRange(KLastElement + 1,11,1,1).setValue(average); //K
  sheet.getRange(KLastElement + 1,12,1,1).setValue(average); //L

  // Clean I
  sheet.getRange("I2:I").clear();
}
```

A.13 Code on Google Scripts to create an API for sending emails

```
function doGet(e)
{
  // Email:
  var to = e.parameter.to;
  var subject = e.parameter.subject;
  var body = e.parameter.body;

  // Particle Call Function
  var particleFunction = e.parameter.particleFunction;
  var particleValue = e.parameter.particleValue;
  if(to != undefined && subject != undefined && body != undefined)
  {
    MailApp.sendEmail(to, subject, body);
  }
  else if(particleFunction != undefined && particleValue != undefined)
  {
    executeFunctionFromParticle(particleFunction, particleValue);
  }
}

//////////////////////////////////////////////////////////////////// Particle Cloud Function Caller Function
\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
function executeFunctionFromParticle(func, value)
{
  var url = "https://api.particle.io/v1/devices/" + deviceId + "/" + func + "?access_token=" + accessToken;
```

```
  var formData = {
   'arg': value
  };
  var options = {
    'method' : 'post',
    'payload' : formData
  };
  var response = UrlFetchApp.fetch(url, options);
  //Logger.log(response);

  var params = JSON.parse(response.getContentText());

  return params.return_value;
}
```

A.14 Code on Amazon Lambda for integration of Amazon Alexa

```
/**
 * App ID for the skill
 */
var APP_ID = "amzn1.ask.skill.---"; //replaced with "amzn1.echo-sdk-ams.app.[your-unique-value-here]";

/**
 * Use Particle API JS to access the data
 * @see https://docs.particle.io/reference/javascript/
 */
var Particle = require('particle-api-js');
var particle = new Particle();

/**
 * Particle authentication credentials
 * @see https://docs.particle.io/reference/api/#authentication
 */
var PARTICLE_DEVICE_ID = "---";
var PARTICLE_ACCESS_TOKEN = "---";

/**
 * The AlexaSkill prototype and helper functions
 */
var http = require('https');
var AlexaSkill = require('./AlexaSkill');

/**
 * ParticleSkill is a child of AlexaSkill.
 * To read more about inheritance in JavaScript, see the link below.
 *
 * @see https://developer.mozilla.org/en-US/docs/Web/JavaScript/Introduction_to_Object-
Oriented_JavaScript#Inheritance
 */
var ParticleSkill = function () {
    AlexaSkill.call(this, APP_ID);
};
```

```
// Extend AlexaSkill
ParticleSkill.prototype = Object.create(AlexaSkill.prototype);
ParticleSkill.prototype.constructor = ParticleSkill;

ParticleSkill.prototype.eventHandlers.onSessionStarted = function (sessionStartedRequest, session) {
    console.log("ParticleSkill onSessionStarted requestId: " + sessionStartedRequest.requestId
        + ", sessionId: " + session.sessionId);
    // any initialization logic goes here
};

ParticleSkill.prototype.eventHandlers.onLaunch = function (launchRequest, session, response) {
    console.log("ParticleSkill onLaunch requestId: " + launchRequest.requestId + ", sessionId: " +
session.sessionId);
    var speechOutput = "Welcome to SmartBrain. You can ask me for the programs running on the machine,
utilization, or MTConnect variables.";
    var repromptText = "With SmartBrain you can ask for the programs running on the machine, utilization,
or MTConnect variables.";
    response.ask(speechOutput, repromptText);
};

ParticleSkill.prototype.eventHandlers.onSessionEnded = function (sessionEndedRequest, session) {
    console.log("ParticleSkill onSessionEnded requestId: " + sessionEndedRequest.requestId
        + ", sessionId: " + session.sessionId);
    // any cleanup logic goes here
};

var functionParticle;
ParticleSkill.prototype.intentHandlers = {
    // register custom intent handlers
    "ParticleSkillIntent": function (intent, session, response) {

        var commandSlot = intent.slots.command;
        var command = commandSlot ? intent.slots.command.value : "";

        var sensorSlot = intent.slots.sensor;
        var sensor = sensorSlot ? intent.slots.sensor.value : "";

        var speakText = "";

        if (sensor) {
            var speechOutput = "";
            switch (sensor) {
                case "programs":
                case "program":
                case "parts":
                case "part":
                    particle.getVariable({ deviceId: PARTICLE_DEVICE_ID, name: "programs", auth:
PARTICLE_ACCESS_TOKEN }).then(function (data) {
                        console.log('Particle variable retrieved successfully: ', data);
                        speechOutput = data.body.result;

                        response.tellWithCard(speechOutput, "SmartBrain", speechOutput);
                    }, function [33] {
                        console.log('An error occurred while getting attrs:', err);
```

```
                var speechOutput = "There was an error when requesting the information";
                response.tellWithCard(speechOutput, "SmartBrain", speechOutput);
            });
            break;

        default:
            break;
    }
  }
  else if (command) {
    var speechOutput = "Coudn't understand what you are looking for.";
    switch (command) {
      case "on":
        speechOutput = "Turned on the servo motor to go back and forth."
        functionParticle = particle.callFunction({ deviceId: PARTICLE_DEVICE_ID, name: "servo",
argument: "on", auth: PARTICLE_ACCESS_TOKEN });
        var postData = "args=" + "on";
        var requestURI = "/v1/devices/" + PARTICLE_DEVICE_ID + "/" + "servo";
        makeParticleRequest(requestURI, postData, function (resp) {
          var json = JSON.parse(resp);
          console.log(command + ": " + json.return_value);
          response.tellWithCard(skillName, invocationName, "Thing is " + command);
        });

        break;
      case "off":
        speechOutput = "Turned off the servo motor."
        functionParticle = particle.callFunction({ deviceId: PARTICLE_DEVICE_ID, name: "servo",
argument: "off", auth: PARTICLE_ACCESS_TOKEN });

        var postData = "args=" + "off";
        var requestURI = "/v1/devices/" + PARTICLE_DEVICE_ID + "/" + "servo";
        makeParticleRequest(requestURI, postData, function (resp) {
          //var json = JSON.parse(resp);
          //console.log(command + ": " + json.return_value);
          //response.tellWithCard(skillName, invocationName, "Thing is " + command);
        });

        break;
      default:
        var degree = Number(command);
        if (degree != NaN) {
          functionParticle = particle.callFunction({ deviceId: PARTICLE_DEVICE_ID, name:
String(degree), argument: command, auth: PARTICLE_ACCESS_TOKEN });

          var postData = "args=" + String(degree);
          var requestURI = "/v1/devices/" + PARTICLE_DEVICE_ID + "/" + "servo";
          makeParticleRequest(requestURI, postData, function (resp) {
            //var json = JSON.parse(resp);
            //console.log(command + ": " + json.return_value);
            //response.tellWithCard(skillName, invocationName, "Thing is " + command);
          });
          speechOutput = "Turned servo motor to " + degree + " degrees";
        }
        else {
          speechOutput = "Coudn't understand what you are looking for.";
```

```
                }
                break;
            }

            response.tellWithCard(speechOutput, "SmartBrain", speechOutput);
        }
        else {
            response.tell("Sorry, I didn't catch what you said");
        }
    },
    "AMAZON.HelpIntent": function (intent, session, response) {
        var speechOutput = "With SmartBrain you can ask for the programs running on the machine,
utilization, or MTConnect variables.";
        response.ask(speechOutput);
    }
};

// Create the handler that responds to the Alexa Request.
exports.handler = function (event, context) {
    // Create an instance of the ParticleSkill skill.
    var particleSkill = new ParticleSkill();
    particleSkill.execute(event, context);
};



function makeParticleRequest(requestURI, postData, callback) {
    var options = {
        hostname: "api.particle.io",
        port: 443,
        path: requestURI,
        method: 'POST',
        headers: {
            'Content-Type': 'application/x-www-form-urlencoded',
            'Authorization': 'Bearer ' + PARTICLE_ACCESS_TOKEN,
            'Accept': '*.*'
        }
    };

    var req = http.request(options, function (res) {
        console.log('STATUS: ' + res.statusCode);
        console.log('HEADERS: ' + JSON.stringify(res.headers));

        var body = "";

        res.setEncoding('utf8');
        res.on('data', function (chunk) {
            console.log('BODY: ' + chunk);
            body += chunk;
        });

        res.on('end', function () {
            callback(body);
        });
    });
```

```
    req.on('error', function (e) {
        console.log('problem with request: ' + e.message);
    });

    // write data to request body
    req.write(postData.toString());
    req.end();
}
```

A.15 Code in Swift for iOS app

```
//
//  ViewController.swift
//  googleSheet
//
//  Created by Mahmoud Parto on 10/23/16.
//  Copyright © 2016 Parto-Tech. All rights reserved.
//

import UIKit


class ViewController: UIViewController, UITableViewDelegate, UITableViewDataSource, XMLParserDelegate
{

    var items: [String] = ["Time Stamp", "Spindle Speed"]
    var details: [String] = ["", ""]
    var timerRefreshData = Timer();

    override func viewDidLoad()
    {
        super.viewDidLoad()

        self.updateDataTimerFunction();
        timerRefreshData = Timer.scheduledTimer(timeInterval: 1.0, target: self, selector:
#selector(self.updateDataTimerFunction), userInfo: nil, repeats: true);
    }

    override func viewDidDisappear(_ animated: Bool) {
        timerRefreshData.invalidate()
    }

    override func didReceiveMemoryWarning()
    {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }



    // tableview
    @IBOutlet weak var tableView: UITableView!

    func numberOfSections(in tableView: UITableView) -> Int
    {
        return 1;
    }


    func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int
    {
        return items.count
```

```
}

// name of the headers
func tableView(_ tableView: UITableView, titleForHeaderInSection section: Int) -> String?
{
   return "Current status"
}

// height of the header
func tableView(_ tableView: UITableView, heightForHeaderInSection section: Int) -> CGFloat
{
   return 50
}


// Cell for row
func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell
{

   // Spindlespeed
   let cell:UITableViewCell = self.tableView.dequeueReusableCell(withIdentifier: "cell")! as UITableViewCell

   cell.textLabel?.text = self.items[indexPath.row]
   if self.details.count > indexPath.row
   {
      cell.detailTextLabel?.text = self.details[indexPath.row]
   }

   return cell

}

// Did select row
func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath)
{
   tableView.deselectRow(at: indexPath, animated: true)
}

// timer function to get the data from Googlesheet
func updateDataTimerFunction()
{

   guard let myURL = URL(string: selectedMachineURL) else
   {
      print("Error: \(selectedMachineURL) doesn't seem to be a valid URL")
      // show a message
      let hud = MBProgressHUD.showAdded(to: self.view, animated: true)
      hud?.mode = MBProgressHUDMode.text
      hud?.labelText = "Error: \(selectedMachineURL) doesn't seem to be a valid URL"
      hud?.hide(true, afterDelay: 2)
      return
   }

   do
   {
      let myHTMLString = try String(contentsOf: myURL, encoding: .ascii)
      //print("HTML : \(myHTMLString)")

      var splittedXML = myHTMLString.components(separatedBy: "<")

      // Get time stamp
      var splittedXMLtimeStamp = myHTMLString.components(separatedBy: "\"")
      var timestamp = ""
      for var i in 0  ..< splittedXMLtimeStamp.count
      {
         if splittedXMLtimeStamp[i].contains("creationTime")
```

```
                {
                    timestamp = splittedXMLtimeStamp[i+1]
                    print("timestamp:" + timestamp)
                    break
                }
            }

            // get spindle speed
            var spindleSpeed = ""
            for var i in 0  ..< splittedXML.count
            {
                if splittedXML[i].contains("SpindleSpeed ")
                {
                    spindleSpeed = splittedXML[i].components(separatedBy: ">")[1]
                    print("spindlespeed:" + spindleSpeed)
                    break
                }
            }

            // if you couldn't find the spindle speed, look for AngularVelocity
            var angularVelocities:[String] = []
            if spindleSpeed == ""
            {
                for var i in 0  ..< splittedXML.count
                {
                    if splittedXML[i].contains("RotaryVelocity ")
                    {
                        angularVelocities.append(splittedXML[i].components(separatedBy: ">")[1])
                    }
                }
            }

            // write them in the table
            if spindleSpeed != ""
            {
                items = ["Time Stamp", "Spindle Speed"]
                details = [timestamp, spindleSpeed]
            }
            else
            {
                items = ["Time Stamp"]
                details = [timestamp]
                for var i in 0 ..< angularVelocities.count
                {
                    items.append("Rotary Velocity")
                    details.append(angularVelocities[i])
                }
            }

            tableView.reloadData()


        }
        catch let error
        {
            // print("Error: \(error)")
            // show a message
            let hud = MBProgressHUD.showAdded(to: self.view, animated: true)
            hud?.mode = MBProgressHUDMode.text
            hud?.labelText = "Error: \(selectedMachineURL) doesn't seem to be a valid URL"
            hud?.hide(true, afterDelay: 2)
            return
        }
        self.tableView.reloadData()
    }
}
```

```
//
//  MachineSelectionTableViewController.swift
//  GATech-MTC
//
//  Created by Mahmoud Parto on 12/1/16.
//  Copyright © 2016 Parto-Tech. All rights reserved.
//

import UIKit

class MachineSelectionTableViewController: UIViewController, UITableViewDelegate, UITableViewDataSource
{

    @IBOutlet weak var tableView: UITableView!
    var dataTitles = ["MTConnect Sample Agent"]
    var dataDetails = ["http://agent.mtconnect.org/current"]

    var machineNames:[String] = [];
    var machineURLs:[String] = [];

    override func viewDidLoad()
    {
        super.viewDidLoad()

        // Uncomment the following line to preserve selection between presentations
        // self.clearsSelectionOnViewWillAppear = false

        // Uncomment the following line to display an Edit button in the navigation bar for this view controller.
        // self.navigationItem.rightBarButtonItem = self.editButtonItem()
    }

    override func viewWillAppear(_ animated: Bool)
    {
        // First read the data
        let defaultsRead = UserDefaults.standard
        if defaultsRead.stringArray(forKey: keyMachineNames) != nil
        {
            machineNames = defaultsRead.stringArray(forKey: keyMachineNames)!
            machineURLs = defaultsRead.stringArray(forKey: keyMachineURL)!
            dataTitles = ["MTConnect Sample Agent"]
            dataDetails = ["http://agent.mtconnect.org/current"]
            for var i in 0  ..< machineNames.count
            {
                dataTitles.append(machineNames[i])
                dataDetails.append(machineURLs[i])
            }
        }
    }

    override func didReceiveMemoryWarning()
    {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }

    // MARK: - Table view data source
    func numberOfSections(in tableView: UITableView) -> Int
    {
        return 1
    }

    func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int
    {
        return dataTitles.count
    }
```

```
    func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell
    {
        let cell = tableView.dequeueReusableCell(withIdentifier: "cell", for: indexPath)
        cell.textLabel?.text = dataTitles[indexPath.row]
        cell.detailTextLabel?.text = dataDetails[indexPath.row]
        return cell
    }

    func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath)
    {
        // deselect it first
        tableView.deselectRow(at: indexPath, animated: true)

        // save the google id
        selectedMachineURL = dataDetails[indexPath.row]

        // segue to spindle speed page
        performSegue(withIdentifier: "segueFromMainToDataShowing", sender: self)
    }


    // Override to support conditional editing of the table view.
    func tableView(_ tableView: UITableView, canEditRowAt indexPath: IndexPath) -> Bool
    {
        // Return false if you do not want the specified item to be editable.
        return true
    }


    // Override to support editing the table view.
    func tableView(_ tableView: UITableView, commit editingStyle: UITableViewCellEditingStyle, forRowAt indexPath:
IndexPath) {
        if editingStyle == .delete
        {
            // Delete the row from the data source

            // if it's the first one (MTConnect Sample), show can't be deleted
            if(indexPath.row == 0)
            {
                let hud = MBProgressHUD.showAdded(to: self.view, animated: true)
                hud?.mode = MBProgressHUDMode.text
                hud?.labelText = "Sample agent cannot be removed."
                hud?.hide(true, afterDelay: 2)
                tableView.reloadData()
                return
            }

            dataTitles.remove(at: indexPath.row)
            dataDetails.remove(at: indexPath.row)

            var tempDataTitles = dataTitles
            var tempDataDetails = dataDetails
            tempDataTitles.remove(at: 0) // remove the sample webpage of MTConnect
            tempDataDetails.remove(at: 0) // remove the sample webpage of MTConnect

            // store the revised arrays
            let defaults = UserDefaults.standard
            if(tempDataTitles.count > 0)
            {
                defaults.setValue(tempDataTitles, forKey: keyMachineNames)
                defaults.setValue(tempDataDetails, forKey: keyMachineURL)
            }
            else
            {
```

```
            defaults.removeObject(forKey: keyMachineNames)
            defaults.removeObject(forKey: keyMachineURL)
        }
        defaults.synchronize()

        tableView.reloadData()
    }
    else if editingStyle == .insert
    {
        // Create a new instance of the appropriate class, insert it into the array, and add a new row to the table view
    }
}


/*
// Override to support rearranging the table view.
override func tableView(_ tableView: UITableView, moveRowAt fromIndexPath: IndexPath, to: IndexPath) {

}
*/

/*
// Override to support conditional rearranging of the table view.
override func tableView(_ tableView: UITableView, canMoveRowAt indexPath: IndexPath) -> Bool {
    // Return false if you do not want the item to be re-orderable.
    return true
}
*/

/*
// MARK: - Navigation

// In a storyboard-based application, you will often want to do a little preparation before navigation
override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
    // Get the new view controller using segue.destinationViewController.
    // Pass the selected object to the new view controller.
}
*/

}
//
//  AddMachineViewController.swift
//  GATech-MTC
//
//  Created by Mahmoud Parto on 12/1/16.
//  Copyright © 2016 Parto-Tech. All rights reserved.
//

import UIKit

class AddMachineViewController: UIViewController
{

    @IBOutlet weak var machineNameTextField: UITextField!
    @IBOutlet weak var machineGoogleScriptsIDTextField: UITextField!

    override func viewDidLoad()
    {
        super.viewDidLoad()
    }

    var machineNames:[String] = [];
    var machineURLs:[String] = [];

    override func viewWillAppear(_ animated: Bool)
    {
```

```
    // First read the data
    let defaultsRead = UserDefaults.standard

    if defaultsRead.stringArray(forKey: keyMachineNames) != nil
    {
        machineNames = defaultsRead.stringArray(forKey: keyMachineNames)!
        machineURLs = defaultsRead.stringArray(forKey: keyMachineURL)!
    }
}

override func didReceiveMemoryWarning()
{
    super.didReceiveMemoryWarning()
    // Dispose of any resources that can be recreated.
}

@IBAction func doneButtonClicked(_ sender: Any)
{
    let machineName = machineNameTextField.text;
    let machineURL = machineGoogleScriptsIDTextField.text;

    // if they were empty don't go further
    if (machineName == "" )
    {
        // show a message
        let hud = MBProgressHUD.showAdded(to: self.view, animated: true)
        hud?.mode = MBProgressHUDMode.text
        hud?.labelText = "Machine name cannot be empty"
        hud?.hide(true, afterDelay: 2)
    }
    else if (machineURL == "" )
    {
        // show a message
        let hud = MBProgressHUD.showAdded(to: self.view, animated: true)
        hud?.mode = MBProgressHUDMode.text
        hud?.labelText = "URL cannot be empty"
        hud?.hide(true, afterDelay: 2)
    }
    else
    {
        // if all is fine, save it and go back to the main
        let defaults = UserDefaults.standard

        machineNames.append(machineName!);
        machineURLs.append(machineURL!);

        // store the revised arrays
        defaults.setValue(machineNames, forKey: keyMachineNames)
        defaults.setValue(machineURLs, forKey: keyMachineURL)
        defaults.synchronize()

        performSegue(withIdentifier: "segueFromAddMachineToMain", sender: self)

    }

}
@IBAction func backButtonClicked(_ sender: Any)
{
    performSegue(withIdentifier: "segueFromAddMachineToMain", sender: self)
}


/*
// MARK: - Navigation

// In a storyboard-based application, you will often want to do a little preparation before navigation
```

```
    override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
        // Get the new view controller using segue.destinationViewController.
        // Pass the selected object to the new view controller.
    }
    */

}
```

6. REFERENCES

[1] L. E. S. Oliveira and A. J. Álvares, "Axiomatic Design Applied to the Development of a System for Monitoring and Teleoperation of a CNC Machine through the Internet," *Procedia CIRP*, vol. 53, pp. 198-205, 2016.

[2] R. Y. Zhong, Q. Y. Dai, T. Qu, G. J. Hu, and G. Q. Huang, "RFID-enabled real-time manufacturing execution system for mass-customization production," *Robotics and Computer-Integrated Manufacturing*, vol. 29, no. 2, pp. 283-292, 2013.

[3] R. Y. Zhong, S. Lan, C. Xu, Q. Dai, and G. Q. Huang, "Visualization of RFID-enabled shopfloor logistics Big Data in Cloud Manufacturing," *The International Journal of Advanced Manufacturing Technology*, vol. 84, no. 1-4, pp. 5-16, 2015.

[4] R. Y. Zhong, L. Wang, and X. Xu, "An IoT-enabled Real-time Machine Status Monitoring Approach for Cloud Manufacturing," *Procedia CIRP*, vol. 63, pp. 709-714, 2017.

[5] J. Lee, E. Lapira, B. Bagheri, and H.-a. Kao, "Recent advances and trends in predictive manufacturing systems in big data environment," *Manufacturing Letters*, vol. 1, no. 1, pp. 38-41, 2013.

[6] G. Chryssolouris, D. Mavrikios, N. Papakostas, D. Mourtzis, G. Michalos, and K. Georgoulias, "Digital manufacturing: History, perspectives, and outlook," *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, vol. 223, no. 5, pp. 451-462, 2009.

[7] (November 23, 2010, October 2017). *RS-232 vs. TTL Serial Communication*. Available: https://www.sparkfun.com/tutorials/215

[8] J.-C. Cano, J.-M. Cano, E. González, C. Calafate, and P. Manzoni, "Power characterization of a bluetooth-based wireless node for ubiquitous computing," in *Wireless and Mobile Communications, 2006. ICWMC'06. International Conference on*, 2006, pp. 13-13: IEEE.

[9] P. Mohan, V. N. Padmanabhan, and R. Ramjee, "Nericell: rich monitoring of road and traffic conditions using mobile smartphones," in *Proceedings of the 6th ACM conference on Embedded network sensor systems*, 2008, pp. 323-336: ACM.

[10] T. Pering, Y. Agarwal, R. Gupta, and R. Want, "Coolspots: reducing the power consumption of wireless mobile devices with multiple radio interfaces," in *Proceedings of the 4th international conference on Mobile systems, applications and services*, 2006, pp. 220-232: ACM.

[11] X. Zhang and G. Riley, "An on-demand bluetooth scatternet formation and routing protocol for wireless sensor networks," in *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, 2005 and First ACIS International Workshop on Self-Assembling Wireless Networks. SNPD/SAWN 2005. Sixth International Conference on*, 2005, pp. 411-418: IEEE.

[12] J. Postel, "Transmission control protocol," 1981.

[13] R. Braden, "Requirements for Internet hosts-communication layers," 1989.

[14] D. D. Clark, "Window and acknowledgement strategy in TCP," 1982.

[15] D. D. Clark, "The design philosophy of the DARPA internet protocols," *ACM SIGCOMM Computer Communication Review*, vol. 25, no. 1, pp. 102-111, 1995.

[16] D. Comer, "Internetworking with TCP/IP, Vol. I: Principles, Protocols, and Architecture, 3/e," ed: Englewood Cliffs (NJ): Prentice-Hall, 1995.

[17] K. R. Fall and W. R. Stevens, *TCP/IP illustrated, volume 1: The protocols*. addison-Wesley, 2011.

[18] V. Jacobson, "Congestion avoidance and control," in *ACM SIGCOMM computer communication review*, 1988, vol. 18, no. 4, pp. 314-329: ACM.

[19] V. Jacobson, "Modified TCP congestion avoidance algorithm," *end2end-interest mailing list*, 1990.

[20] P. Srisuresh and K. Egevang, "Traditional IP network address translator (Traditional NAT)," 2070-1721, 2000.

[21] (October 2017). *UDP and TCP, two ways of sending traffic*. Available: https://www.homenethowto.com/ports-and-nat/udp-and-tcp-two-ways-of-sending-traffic/

[22] W. Colitti, K. Steenhaut, N. De Caro, B. Buta, and V. Dobrota, "REST Enabled Wireless Sensor Networks for Seamless Integration with Web Applications," presented at the 2011 IEEE Eighth International Conference on Mobile Ad-Hoc and Sensor Systems, 2011.

[23] M. Masse, *REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces*. " O'Reilly Media, Inc.", 2011.

[24] A. Foster. (2015, October 2017). *Juggling Data Connectivity Protocols for Industrial IoT*. Available: https://www.eetimes.com/author.asp?section_id=36&doc_id=1326169&

[25] S. Atluru and A. Deshpande, "Data to information: can MTConnect deliver the promise," *Transactions of NAMRI/SME*, vol. 37, no. 2009, pp. 197-204, 2009.

[26] R. Lynn, W. Louhichi, M. Parto, E. Wescoat, and T. Kurfess, "Rapidly deployable MTConnect-based machine tool monitoring systems," in *ASME 2017 12th International Manufacturing Science and Engineering Conference collocated with the JSME/ASME 2017 6th International Conference on Materials and Processing*, 2017, pp. V003T04A046-V003T04A046: American Society of Mechanical Engineers.

[27] A. Vijayaraghavan, "MTConnect for realtime monitoring and analysis of manufacturing enterprises," in *Proceedings of the international conference on digital enterprise technology, Hong Kong*, 2009.

[28] L. Wang, P. Orban, A. Cunningham, and S. Lang, "Remote real-time CNC machining for web-based manufacturing," *Robotics and Computer-Integrated Manufacturing*, vol. 20, no. 6, pp. 563-571, 2004.

[29] P. Waurzyniak, "Electronic intelligence in manufacturing," *Manufacturing Engineering*, vol. 127, no. 3, pp. 44-44, 2001.

[30] (October 2017). *MTConnect is the communication standard of choice for manufacturing.* Available: http://www.mtconnect.org/

[31] Y.-K. Chen, "Challenges and opportunities of internet of things," in *Design Automation Conference (ASP-DAC), 2012 17th Asia and South Pacific*, 2012, pp. 383-388: IEEE.

[32] G. Pardo-Castellote, "Omg data-distribution service: Architectural overview," in *Distributed Computing Systems Workshops, 2003. Proceedings. 23rd International Conference on*, 2003, pp. 200-206: IEEE.

[33] J. Kristoff *et al.*, "This is a plagiarism that combines parts ripped out from (at least) the following three works," *ACM SIGCOMM Computer Communication Review*, vol. 40, p. 3, 2010.