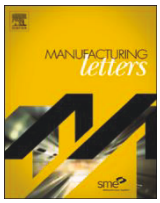
Available online at www.sciencedirect.com

Manufacturing Letters

Manufacturing Letters 00 (2022) 000–000



50th SME North American Manufacturing Research Conference (NAMRC 50, 2022)

Implementing an open-source sensor data ingestion, fusion, and analysis capabilities for smart manufacturing

Kerry Wang[a], Parth Dave[b], Abhishek Hanchate[b], Dinakar Sagapuram[b], Gautam Natarajan[b], Satish T.S. Bukkapatnam[b*]

[a] *College of William and Mary, Department of Computer Science, Sadler Center, 200 Stadium Dr, Williamsburg, VA 23185, USA*
[b] *Texas A&M University, Department of Industrial and Systems Engineering, 3131 TAMU, College Station, TX 77843, USA*

* Satish T.S. Bukkapatnam. Tel.: +1-979-458-2348; *E-mail address:* satish@tamu.edu

Abstract

In many modern enterprises, factory managers monitor their machinery and processes to prevent faults and product defects, and maximize the productivity and efficiency. Asset condition, product quality and system productivity monitoring consume some 40-70% of the production costs. Oftentimes, resource constraints have prevented the adoption and implementation of these practices in small businesses. Recent evolution of manufacturing-as-a-service and increased digitalization opens opportunities for small and medium scale companies to adopt smart manufacturing practices, and thereby surmount these constraints. Specifically, sensor wrappers that delineate the specifications of sensor integration into manufacturing machinery, with appropriate edge-cloud computing and communication architecture can provide even small businesses with a real-time data pipeline to monitor their manufacturing machines. However, the data in itself is difficult to interpret locally. Additionally, proprietary standards and products of the various components of a sensor wrapper make it difficult to implement a sensor wrapper schema. In this paper, we report an open-source method to integrate sensors into legacy manufacturing equipment and hardware. We had implemented this pipeline with off-the-shelf sensors to a polisher (from Buehler), a shaft grinding machine (from Micromatic), and a hybrid manufacturing machine (from Optomec), and used hardware and software components such as a National Instruments Data Acquisition (NI-DAQ) module to collect and stream live data. We evaluate the performance of the data pipeline as it connects to the Smart Manufacturing Innovation Platform (SMIP)—web-based data ingestion platform part of the Clean Energy Smart Manufacturing Innovation Institute (CESMII), a U.S. Department of Energy-sponsored initiative—in terms of data volume versus latency tradeoffs. We demonstrate a viable implementation of Smart Manufacturing by creating a vendor-agnostic web dashboard that fuses multiple sensors to perform real-time performance analysis with lossless data integrity.

Keywords: smart manufacturing; sensor fusion; open-source; real-time analysis

1. Introduction and background

In recent years, new business models are emerging to support custom manufacturing to meet the personal preferences of individual customers [1-3], ostensibly without compromising on the efficiencies and functionality [4]. While the technological advances during the past century have enhanced the efficiencies and performance of high-volume manufacturing systems, manufacturing of low-volume, high-mix custom products tends to incur comparatively longer cycle times, higher unit cost, and lower productivity [3]. These issues with custom manufacturing, together with the advent of technologies such as additive/hybrid manufacturing, cloud computing, sensor fusion, and IoT are fueling a trend towards digitization and the emergence of a manufacturing-as-a-service (MaaS) paradigm [5]. This paradigm promotes a democratized manufacturing sector where the customers as well as the major firms and producers alike can "uber" their (custom) production requirements to a distribution of smart micro- small- and medium-enterprises (MSMEs) [5]. Moreover, such digitization

and democratization of the manufacturing industry promotes original equipment manufacturing (OEM) and in-turn enhances innovation [6-10]. The presence of MSMEs in the cloud will keep them competitive and they can gain same level of visibility in the cyberspace as the big firms [11]. The companies that take a step towards virtualizing their business models and operations are likely to be among the first receivers of benefits from democratized manufacturing environment.

Pertinently, MSMEs currently constitute over 90% of modern manufacturers, and a vast majority of them face significant challenges, the so-called digital divide barrier, in adopting smart manufacturing (SM) practices [12][5]. These challenges impede their entry into the MaaS paradigm [5]. In this context, over 70% of the manufacturing equipment, especially of those in MSMEs, are legacy machines. Resource constraints often prevent the upgrade and replacement of old machinery with those capable of connecting to the digital thread and leverage the on-line solutions to enhance their equipment condition, quality, and productivity [13].

It is now becoming increasingly feasible to integrate the current machines and other assets of a manufacturing system with commercial off-the-shelf (COTS) sensors and industrial IOT technology as a part of an SM platform. However, mere sensorization does not meet the critical SM requirement of gaining access to reliable, accurate, and up-to-date data. The data needs to be collected, stored, and accessed while preserving its integrity. Additionally, it should be analyzed to provide timely and meaningful insights and inputs [2].

In such a data processing pipeline, there must first exist an agreed-upon protocol to transfer information. Unfortunately, the manufacturing industry is currently inundated with competing SM related standards and closed architecture products [14]. A mixed-vendor stack must deal with incompatible processes and data structures, undermining SM's focus on cooperation and plug-and-play [15]. Sophisticated and complete package solutions such as OSIsoft's PI System exist that overcome this limitation. However, many of these are proprietary and expensive for adoption into many MSME environments. For example, in 2015, the city of Holland, Michigan, was quoted $118,968 for a PI System deployment at Holland Energy Park, a natural gas power plant [16].

A few computational platforms, such as the Smart Manufacturing Innovation Platform (SMIP) of the Clean Energy Smart Manufacturing Innovation Institute (CESMII) [17] have emerged to address this limitation. A few parallel efforts that have been made towards sensorizing legacy machines, sending collected data to a database, analyzing it, and displaying it at client's end as a webpage, exist in literature [18-20]. Some of these implementations have investigated the introduction of different IoT elements, edge and cloud-computing architectures, and machine learning methods. For example, Verma *et al.* [21] used a case study of an IoT-based vibration monitoring using commercial off-the-shelf sensors with open-source connections. However, the earlier methods largely ignore the data pipelines and the sensor wrapper needed to integrate an MSME as part of an established smart manufacturing platform, such as OSIsoft PI or SMIP. Again, a systematic study of data integrity and latency issues in a realistic context that considers a distributed, multivendor, and multi-ownership scenario does not exist.

The proposed work differs from the prior efforts in studying the issues of accuracy and transmission of the data in the SM context that consider multivendor and multi-owner hardware and software data pipelines to integrate a manufacturer with an SM platform. The specific focus will be on how the data integrity and latency emerge in such real-world scenarios. These considerations can offer an approach to benchmark the performance of viable SM implementations.

Moreover, the SM architecture described in this paper and its capabilities are unique in terms of its plug-and-play nature, real-time signal visualization and analytics, provision for various desirable metrics, and fusion of multiple sensors. It can achieve high throughput streaming with availability of open-source connections, and can integrate with CESMII's applications marketplace which is akin to an "App Store" where developers of such SM based applications can make their products available to customers via SMIP. Botcha *et al.* [2] provides the workflow towards transforming the current manufacturing systems into SM platform-integrated environments. This paper extends that work on integrating a traditional manufacturing environment with an SM platform by demonstrating a viable SMIP implementation built entirely on an open-source stack.

We developed a viable implementation of SM which is both completely open-source and also compliant with a reliable cloud historian that can handle high frequency data. A live analysis functionality was also a part of this implementation and comes with our own web dashboard. The dashboard displays the real-time data stream in the time-domain, frequency-domain (FFT), as well as the time-frequency domain (spectrogram). It has capabilities for integrating desired add-in applications or metrics and can handle any further analyses based on these.

For the SM implementation's use-case on CESMII's SMIP built based on the aforementioned investigations, we were able to preserve data integrity and achieve data loss-free transmission, i.e., the SMIP database did not drop even one sample of data for up to 23 kHz throughput rate. With this high throughput rate, we were able to enable cloud analytics for quality assurance and visualization. These capabilities also expand the possible set of applications and enhances the scalability and reproducibility of our SM implementation.

This paper is organized as followed. Section 2 delves into the methodology and implementation. The architecture used, communication channel, dashboard development, and testing are also discussed here. Results and findings are summarized in Section 3 with extensive discussion on performance. Section 4 summarizes the paper with a few conclusions based on our implementation and testing.

2. Methodology and implementation

2.1. System architecture

Our system architecture is loosely based on CESMII's model [16] and is composed of three components as shown in figure 1: Edge Adapters, Data Historian, and Analytics Modules.

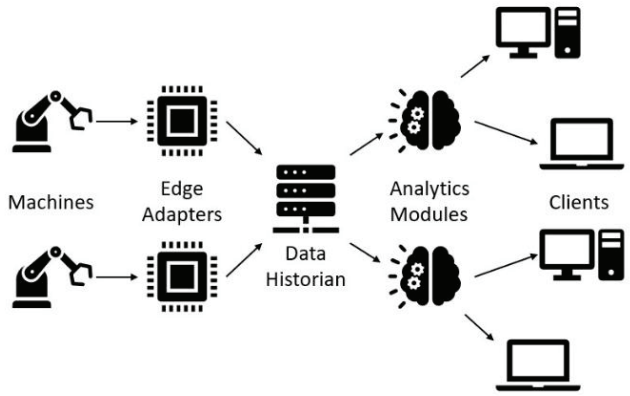


Figure 1. Diagram of the system architecture

The Machines in Figure 1 correspond to a group of equipment on which various sensors are mounted for data acquisition. Edge Adapters physically connect to sensors, poll the sensors for data, and preprocess the data for transport to the Data Historian. The Data Historian stores incoming data from Edge Adapters in a database and responds to Analytics Modules' requests for data. Lastly, Analytics Modules provide user interfaces where users can request analysis. The Analytics Modules then pull relevant data from the Data Historian, performs the analysis, and display the results to the user/client. The Clients have the ability to specify certain parameters such as the sampling rate and get a visual illustration of real-time analytics given by the Analytics modules. This architecture allows for seamless plug-and-play for various commercial/off-the-shelf implementations of each of these modules.

In this paper, we discuss various connections we have implemented in our SM architecture described in figure 2. The proposed SM architecture has 4 layers in it which includes data collection, data processing and edge-based analytics, cloud-based storage and analytics, and anyplace visualization. In the first layer, multimodal data is collected from various sensors by utilizing suitable DAQ systems which are connected to an edge device such as a local computer. The connection between the sensors and DAQ systems is established via BNC cables while that between DAQ systems and the edge device is achieved via USB cables. For the second layer, several options are available for analyzing the collected data on the edge device. This layer harnesses the power of various machine learning techniques to process the data, conduct edge-based analyses such as those in the time domain, frequency domain, and time-frequency domain in form of real-time time series, FFT, and spectrogram plots.

Similarly, this second layer is also capable of hosting applications such as predictive models which can be utilized to predict desired parameters and metrics. The processed data and supplementary information from the hosted applications can then be uploaded into the third layer with transmission usually via ethernet or LAN cables. This layer acts as a data historian for common data management and historization purposes with enabled browsing of data entities. Followed by this, the architecture also has capabilities for live real-time analysis of the data using an interactive dashboard as a part of the fourth layer.

The proposed SM architecture is unique in its plug-and-play nature wherein various options are available at each of the layers discussed above. The user can install and integrate additional or replace already existing COTS components in the four layers as per their preference. These add-on components in the architecture can be both proprietary and open-source. There is also freedom to host additional desired applications and metrics in various locations in the existing architecture. Moreover, such a plug-and-play nature allows for potential edge-cloud partitioning and aiding the achievement of a sweet spot between computational requirements and expenditure. At this stage, we have successfully implemented the routes highlighted by green arrows in figure 2. At layer one, we have

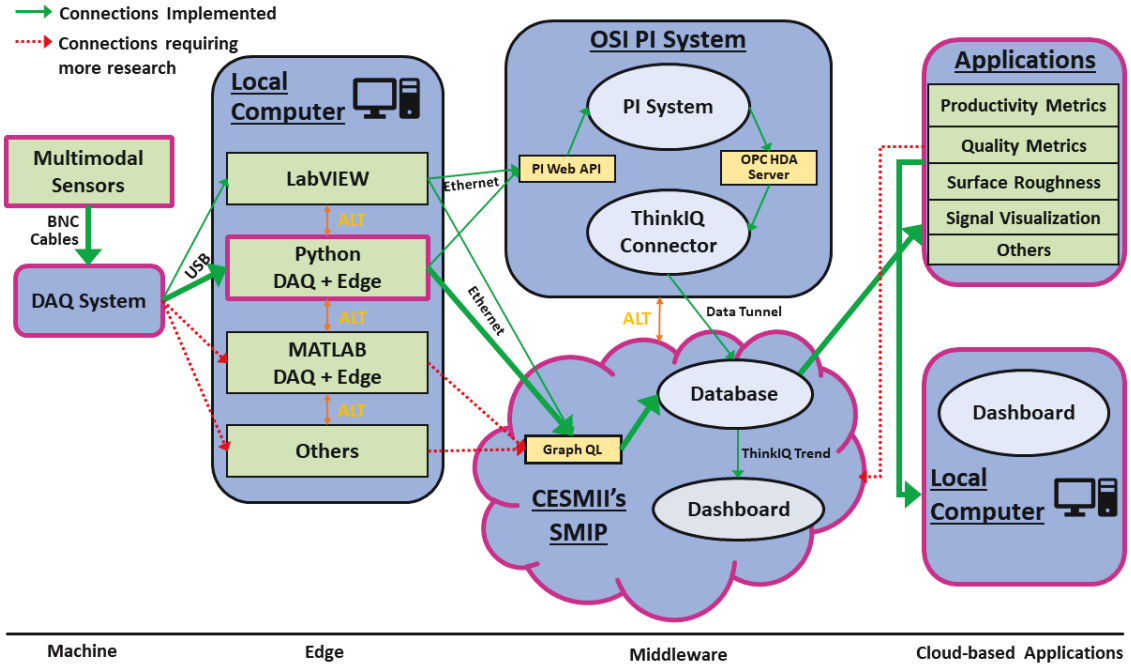


Figure 2. Plug-and-play architectures lets Analytics Modules bring in any tools they need

sensors such as accelerometer, acoustic emissions sensor, and force sensor, and we collect the data from them using National Instrument (NI) DAQs. Analytics and data processing options such as LabVIEW, Python, and MATLAB based implementations exist in the second layer. Followed by this, we have successfully implemented the third layer via OSIsoft's PI system as well as an open-source GraphQL based cloud historian, CESMII's SMIP.

Our dashboard only relies on edge-based applications such as productivity metrics, quality metrics, and surface roughness predictor as of now at the fourth layer. Our implementation has dashboards which are based on LabVIEW as well as Python. Such a dashboard can be viewed both on the cloud as well as on edge devices, irrespective of the location of these systems and usually on client's end. The routes highlighted by red arrows are still under progress as a part of our SM innovation initiative. In this paper, we will propose and focus on the open source implementation of SM via Python based approach which is highlighted in magenta and thicker green arrows in figure 2. After collecting multimodal sensor data using NI DAQs, the analyses and processing is done via Python. We also have edge-based Python generated metrics as mentioned in figure 2. Followed by this, the processed data is transmitted to CESMII's SMIP in form of JSON objects over the internet. Finally, the data can be downloaded from the SMIP and visualized in the form of an interactive web-based Python dashboard.

2.1.1. Edge adapters

Off-the-shelf sensors commonly output their readings with a simple analog voltage. These signals cannot be used directly, so we must first convert them into digital values. Our implementation uses NI DAQ modules. National Instruments offers a wide variety of DAQ modules with varying tolerances and precisions. These modules have internal analog to digital converters and connect to a computer via USB. Using Python and the nidaqmx library, we can read the digital data and store it in Python lists, a standard data structure in data science.

The storage can also be implemented using NI's Technical Data Management Streaming (TDMS) file format [22], a proprietary high-efficiency format for compatibility with existing analysis workflows. These TDMS files are easy to extract using Python and the npTDMS library. Timestamps of when each sample is taken are also recorded so recreation of the signal is possible later on. The samples along with their timestamps can then be packaged into a JSON object, a common format for transmitting structured data over the internet, compressed, and sent to the Data Historian. An example of such a JSON object can be found in Listing 1. With this approach, Edge Adapters can be deployed on low-cost, low-performance hardware such as Raspberry Pis or internet-enabled microcontrollers because they do not have to perform any computationally intensive analysis.

2.1.2. Data historian

As Edge Adapters and Analytics Modules may be spread across distant networks, the Data Historian must be deployed in the cloud. We chose CESMII's open-source SMIP for this role. The SMIP is a solution by CESMII to deliver industrial plug-and-play framework to the discrete, hybrid, and process manufacturing industries. With secure connectivity to equipment and processes, it adds valuable context with a goal to access information intelligently and in an automated manner. As a part of the SMIP platform, CESMII is also developing the Profiles standards to describe sensing elements, equipment, and processes. It also provides a semantics aspect to the data and describes the relationship among them. The culmination of all these aspects is expected to form the holy grail with an ability to define new systems without extensive middleware reconfigurations and maintenance issues. Specifically, we make use of the fact that SMIP stores data in an internal database which indexes data and allows for fast lookup and retrieval. It also provides a web endpoint, a web address where applications can send requests to insert and retrieve data. In our configuration, SMIP was deployed on a Linux server. It used PostgreSQL as its internal database, and PostGraphile to provide an endpoint. Requests to the endpoint are written in GraphQL, a query language developed by Facebook [23].

2.1.3. Analytics modules

The fundamental aspect of smart manufacturing is data analysis. After collecting data and storing it in an easily accessible location, the Analytics Modules perform the analyses and display the results. Examples of Analytics Modules can be something as simple as a history viewer for raw sensor data or as complex as a live dashboard that outsources data processing to cloud services. We created a web dashboard for this project using Dash, a Python framework for creating data science web apps. Our dashboard runs directly in the web browser without the need of additional software. Moreover, it can be run by the client at any location irrespective of whether they have the software or not since all they have to do is modify the IP on which the dashboard will be displayed and then the user can view it.

```
{
  "query" : `
        mutation AddData($id: BigInt, $entries:
[TimeSeriesEntryInput]) {
          replaceTimeSeriesRange(
              input : {
                  attributeOrTagId : $id ,
                  entries: $entries
              }
          } {
              Json
          }
    `,
    "variables" : {
        "id" : 5356,
        "entries" : [
        {
            "value" : 3.9729e-23 ,
            "timestamp" : "2021-07-25T23:11:41.462553+00:00",
            "status" : 0
        } ,
        {
            "value" : 3.9729e-23 ,
            "timestamp" : "2021-07-26T19:09:57.809348+00:00",
            "status" : 0
        }
      ]
  }
}
```

Listing 1. Example of a JSON object used to upload data

2.2. Implementation of communication library

We achieve seamless communication between components in the SM network with a custom library, which takes care of low-level operations like authentication, managing HTTP sessions, and converting between Python data structures and GraphQL. Our communication library provides a developer-friendly interface for communicating between programs and the SMIP. Communication with SMIP takes place via HTTP POST requests, each containing a JSON object with two keys: query and variables. The query value is a GraphQL query for a database operation, such as adding data or getting data. Each query is a constant string, with placeholder tags for variables. The variables value contains key-value pairs of variables to be substituted into the placeholder tags in the query. Database-modifying requests additionally must include an authentication token. To obtain a token, we must first request a challenge from the endpoint with a username. The SMIP replies with a challenge for the user. We then request a token by sending the challenge combined with the password.

To streamline this process as much as possible for developers, we take an object-oriented approach. The library defines one class that acts as a connection manager. The class constructor takes the endpoint address, a username, and a password as arguments, requests a token, and stores it in the object as a private variable. It also opens an HTTP session to reduce overhead on repeated requests. The class also provides methods for database operations that take Python data types. These methods then construct the appropriate HTTP POST request, attach the token, and return with the server's response. With this approach, how the requests are made is abstracted away from the developer, and database operations can be made with one line of code. A flowchart of this process is shown in figure 3.

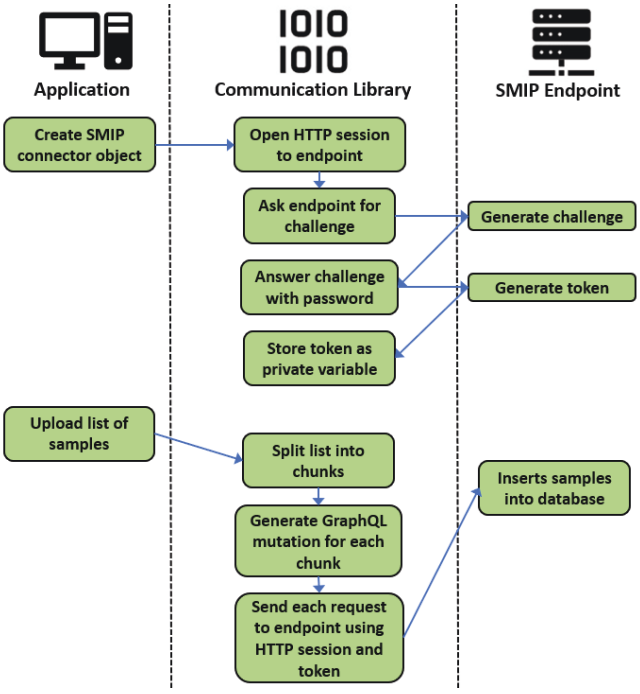


Figure 3. Flowchart of using the communication library to upload data to SMIP

2.3. Implementation of dashboard

The dashboard consists of a web app frontend and a Dash backend. The web app frontend runs in the browser, and the Dash backend is deployed on a server in the cloud. Variables are stored in the browser, and calculations take place on the backend via callbacks. This split of responsibilities is so that any server instance can service any client. Thus the dashboard can be scaled up by simply adding more server instances. Callbacks are functions that the frontend can request the backend to run. In our dashboard, one main callback asks the backend for new data every second, and other callbacks trigger when the data updates. These other callbacks use the updated data to draw new graphs and calculate performance metrics. Since callbacks are self-contained functions, new functionality can be added easily without affecting existing functionality.

A screenshot of the dashboard is shown in figure 4. For this example, we envision a use-case for a grinding machine. For the use-case discussed [24], experiments were conducted on an external cylindrical plunge grinding machine from Micromatic Grinding Technologies Ltd., India. The machine is capable of producing parts with an IT3 tolerance grade by using an A80-L5-V alumina grinding wheel. An EN-31 cylindrical steel workpiece with a hardness value of 60 HRC and initial diameter of 21 mm was used as the starting stock for all experiments. In total, each workpiece goes through four stages, namely, roughing, semi-finishing, finishing, and spark-out. Surface roughness measurements are taken at the end of each stage using a Mahr perthometer. In each stage, data was collected simultaneously from three sources, namely from two accelerometers, one in the tangential and other in the normal direction of the wheel, and a power cell to capture the power drawn by the grinding wheel when engaged with the workpiece. The accelerometers are single axis piezoelectric type from Dytran (Specifications- Range: ±10 g, Sensitivity: 500-1000 mV/g) and the power cell is a hall effect based power cell from Loadcontrol Inc., USA (Specifications- Range: 0-25 kW, Output: 0-10 V/set power range).

The data collection was possible by using National Instruments Data Acquisition Systems- NI DAQ-9234 and NI DAQ-9205 with sampling frequencies of 10 kHz and 66.67 Hz for the accelerometers and the power cell respectively. The process parameters, namely the wheel speed, workpiece speed, and in-feed of the grinding wheel were varied with each experiment.

As seen in figure 4, by monitoring the power usage and the acceleration of the grinding wheel, the dashboard can provide several real-time visualization and insights into the synchronous variations of multiple signal patterns in the time domain, frequency domain, as well as time-frequency domain in form of live time series, FFT, and spectrogram plots. Along with these live visuals, it also captures various metrics.

The dashboard provides a visual and point-and-click interactive web interface based on Python Analytics. It was built using Dash, an open source Python framework written on the top of Flask, Plotly.js, and React.js. Our implementation takes in the data coming from data acquisition systems mentioned above and provides visuals into the real-time time series data signal as well as frequency spectra and other spectral features based on the last second of the streaming data. The

dashboard also has the functionality to select the appropriate window type and other parameters for the generation of spectrograms.

We can keep the performance of our SM implementation in check by tracking the latest update or response from the cloud and the number of data samples received during that time frame. We also have a functionality of tracking the wall time and machine states such as the status of machine being idle or active.

With regards to productivity, by monitoring the power usage, the dashboard can track how long the machine spends actively working versus idling. It can also keep a count of how many parts have been produced which provides useful information and hints the operators on need for optimization of their workflow. It also keeps track of the Run time, Idle time, Down time, and total Elapsed time both in terms of units of time as well as in percentage.

Our dashboard also keeps an indirect track of quality of the parts produced by the machine. The Python and MATLAB based analyses work atop the dashboard in order to perform various real-time analyses and predictions. It keeps a check for any abnormally high power usage and works behind the scenes with an ability to predict and mark these parts as good or potentially anomalous. This provides operators with an early warning of expected malfunctions and allows time for preventative maintenance before any failure. Moreover, by using a pre-trained random forest machine learning regression model in MATLAB, the dashboard can provide the predicted surface roughness value of a produced part using parameters such as feed rate, speed, power usage, and acceleration of the grinding wheel.

The case study of external cylindrical plunge grinding machine demonstrates the SM implementation only for the case of vibration and power cell signals. However, with the growth in imaging technologies, the SM implementation and the predictive models working behind the scenes can be integrated to include image based analyses as well. Although, at present, technologies are available for fast visual inspections, much of the inline imaging technologies which are affordable are confined to measuring and identifying geometrical errors and large morphological defects [25]. They are not mature enough yet to measure aspects such as surface roughness values and other fine features in an industrial environment. In the proposed SM implementation, most predictions and metrics are based on open-source Python based analyses except for the surface roughness which is based on a MATLAB engine based machine learning model. However, it can be easily swapped for a model based on open-source languages such as Python or R. This provides a means of automating the long and complex process of manual inspection of the surface roughness and can result in time and cost savings. Obviously, these can be swapped with for any other use-case with its own desired prediction models and analyses.

The use case provided in this paper is just one example. We envision similar dashboards can be created based on the same framework to monitor multi-sensor data from any machine or a group of machines to look into real-time insights and provide feedback on productivity and quality metrics.

2.4. Testing methodology

To be viable, the SM network must support a high enough bandwidth to provide meaningful analysis and ensure that information is not corrupted or lost during transmission. Since the Data Historian coordinates communication between the network components, we focused our testing on the SMIP and the performance of transmission of data and instructions between the edge and the SMIP.

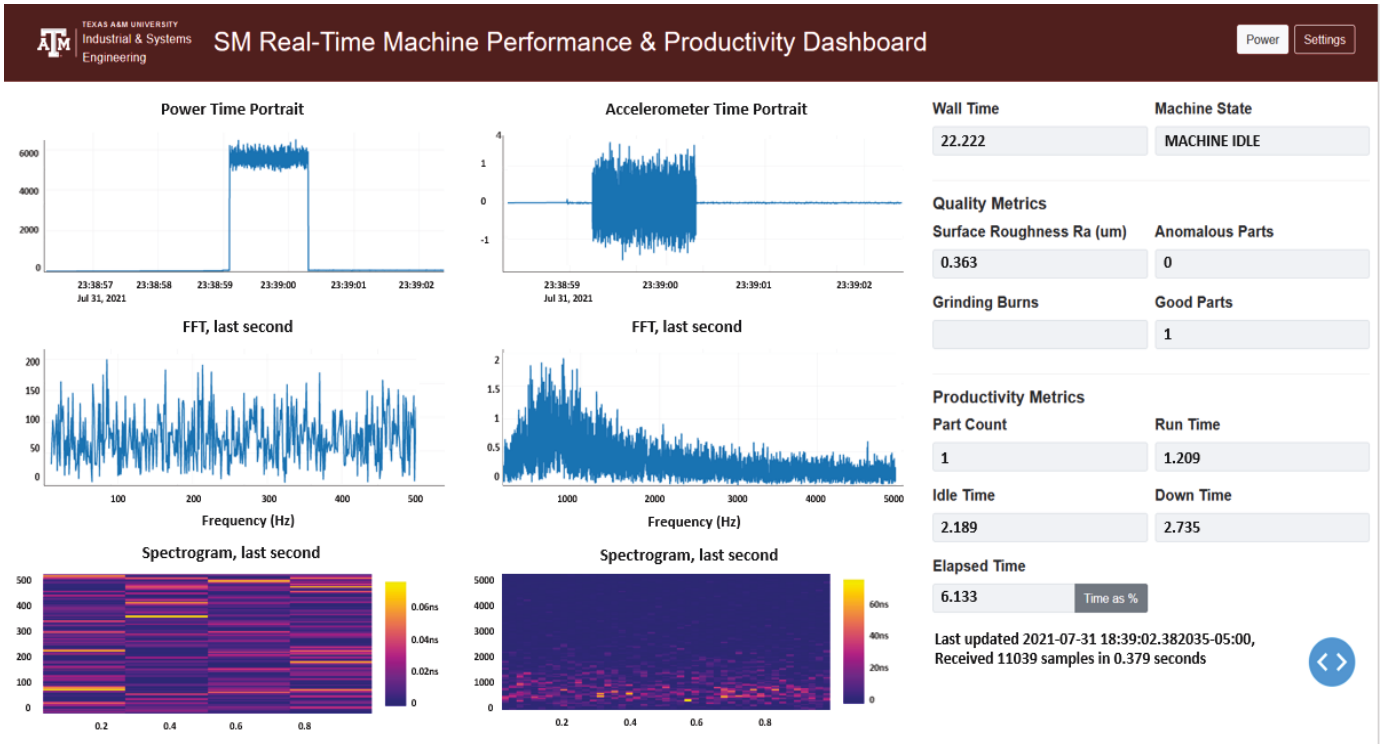


Figure 4. Screenshot of the SM dashboard

To test this, we define Integrity and Speed as our objectives. For integrity, every sample sent must be received. We test for this by creating a dataset of n samples, uploading the dataset to SMIP, downloading the dataset from SMIP, and confirming that the downloaded and original datasets contain the same number of samples. Additionally, the data contained within each sample should not change when transmitted. We tested this by generating a dataset of random-valued samples, uploading it to SMIP, downloading them from SMIP, and calculating the difference between the downloaded dataset and the original. Lastly, speed is tested by timing how long it takes to transfer a dataset of n samples. We tested several different transmission methods. Firstly, SMIP has an upper limit of 8,000 samples per upload request, so larger datasets must be split into chunks before uploading. We tested both small chunk size ($n = 1000$) and large chunk size ($n = 8000$) requests. Secondly, we tested parallelization by comparing the performance of sending requests serially, i.e., one at a time, versus sending them all at once asynchronously.

3. Results and discussion

3.1. Integrity

While testing SMIP's performance, we used a dataset with 240,000 samples between 0 and 1 which were randomly generated. This dataset was uploaded/transmitted to the SMIP database and later on downloaded from the same. It was observed that the database did not drop even a single sample, but there were minor differences in the timestamps and floats. However, as seen in Figures 5 and 6, the differences between the downloaded samples and the original ones are very small. The timestamp differences was uniformly distributed with a maximum and minimum at exactly $\pm 6.0 \times 10^{-7}$ s, respectively, indicating that timestamps are rounded at the microsecond precision. The distribution corresponding to the floating point differences seems to be a combination of multiple gaussian distributions each corresponding to a variable source. The float differences has a mode of 0, indicating that most floats are losslessly preserved. Still, a significant numbers of samples exhibit small variations due to floating-point errors and rounding. However, the size of the differences is not large enough to affect analyses though, so we conclude that SMIP preserves data integrity.

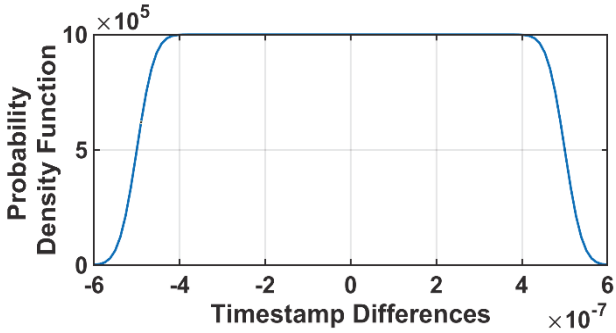


Figure 5. Timestamp difference between downloaded and original data

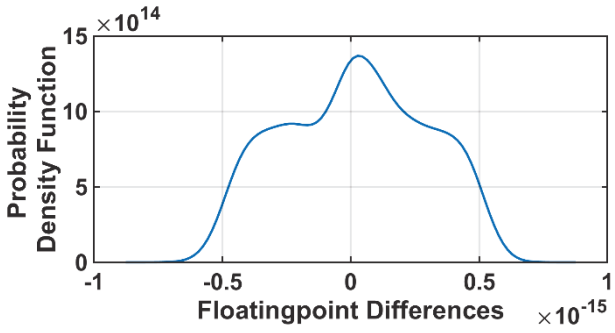


Figure 6. Floating point difference between downloaded and original data

3.2. Speed

As seen in Table 1, we found that when transmitting 240,000 total samples serially, larger chunk sizes significantly increased the performance. Requests using chunks of size 8,000 complete nearly twice as fast as requests using chunks of size 1,000. However, this trend reverses itself once we allow multiple requests at the same time. By using a smaller chunk size, we split one upload task into more numerous requests, which can be more evenly distributed across all server threads.

Table 1. Time to upload 240,000 samples by various methods

Method	Run 1	Run 2	Run 3
Serial 8k	16.881	17.020	16.724
Serial 1k	31.324	31.943	31.350
Async 8k	11.309	10.919	11.114
Async 1k	8.499	8.734	8.808

Having identified asynchronous requests with chunk sizes of 1,000 as the fastest upload method, we measured the time to upload and download increasingly large datasets, the results of which are plotted in Figure 7. We notice that both upload and download trends appear linear and at larger sizes of datasets, the downloads consistently complete faster than uploads. This is expected since the database lookups are much faster than insertions. In the proposed work, the lag consideration due to variations in the internet speeds was not explicitly discussed. In other words, the communication and computing network conditions were treated as random effects as opposed to considering their blocking effects. However, with a faster internet connection with higher upload speeds, we expect the upload trend to be more gradual and behave similar to that of download trend. This behavior is anticipated to change with a weaker internet connection, wherein the upload trend will be much steeper than the download trend.

In our open-source Python based implementation using CESMII's SMIP, we were able to sustain a data rate of around 23,000 samples per second with our configuration, although we expect this will scale by deploying the SMIP database on faster hardware.

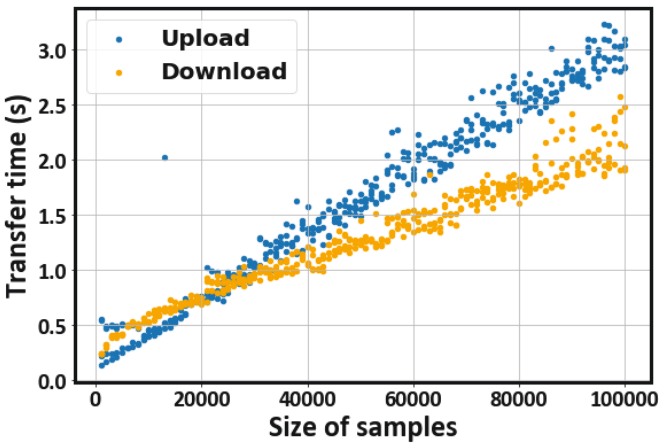


Figure 7. Upload and download times versus sample count. Uploads were async with 1k chunk size.

4. Discussion and concluding remarks

In this project, we tested and demonstrated an open-source implementation that integrates multiple layers of data handling pipeline in SM with CESMII's SMIP.

The effort has also investigated the handling of high-frequency data in SM platform. The open-source implementation of SM provides live visualization and analysis of data via our web dashboard. Our dashboard displays not only the data stream but also data analytics such as FFT and spectrogram along with any add-in applications or metrics such as those based on machine learning predictive models.

Next, the effect of different data upload methods were studied, and the parameters of these were optimized to achieve a loss-free transmission of data at 23 kHz rates. This is one of the highest transmission rates reported with the currently deployed SMIP database and historian. While this data rate per channel of data stream does not match up what was demonstrated with other proprietary based or our OSIsoft PI system implementations, we expect it to scale with deployment of SMIP database over faster hardware and the onset of 5G technologies.

More pertinently, such a high sampling rate is essential to enable advanced process monitoring methods that can learn from IoT vibration and vision sensors combining the computing capabilities at the edge and cloud. We can also measure and analyze certain short time-scale phenomenon associated with a manufacturing process. This opens new doors and helps manufacturers and operators have a better insight into higher frequency features along with a higher quality data stream which was not possible earlier.

Better rate also implies capturing more sample data points for a given timestamp. With the availability of more data, the machine learning models for various metrics can be further improved, making prediction of the results with a better accuracy. The higher data rates also enable functionalities to install additional sensors and measurement challenges, and implement data fusion methods to improve condition monitoring of the machines and quality assurance. Potential applications of our SM implementation include continuous health and productivity monitoring of a machine, incipient detection of anomalous parts from a process on a machine, and anomaly detection schema by looking for any abrupt divergence from the usually observed time and frequency domain plots, among others.

As the data streaming rates improve and the architecture opens to accommodate larger plug-and-play capabilities, the smart manufacturing platforms begin to offer value to the industry, especially to address certain asset management and quality assurance challenges. While the present effort contributes to the advancement of process monitoring in SM platforms, this effort does not address near real-time control of anomalies. Given the latencies, we need to be extremely cognizant of performance of true real-time control however set point control can be achieved based on such cloud based SM systems.

Additionally, these emerging SM architectures and paradigms render the manufacturing environments vulnerable to cyber-attacks [26]. Mahesh *et al.* provides one of the most comprehensive taxonomy in regard to various cyber security threats, attack modes, goals, targets, and preventive countermeasure approaches in this context. In near future, we envision integration of our SM architecture with important cyber security solutions such as secure distributed-streaming of encrypted manufacturing data or instructions [5].

As SMIP matures, we envision this project as a steppingstone for future SM systems, which can build off our work. In the future, we hope to see traditional manufacturing for small- and medium-sized businesses adopt SM to respond to a rapidly developing production landscape. As part of our ongoing effort, we are enhancing the plug-and-play capability with the functionality to choose the desired data pipeline and ingestion option throughout the SM implementation. For example, an OSIsoft PI based dashboard along with Python and MATLAB based data analyses can be combined to offer a custom-solution to an MSME. Such an architecture will provide freedom to the client or user to choose any desired component based on their existing setup without disrupting their operations in any way. CESMII's SMIP is just one use-case among various other SM initiatives. The future efforts include testing of the current implementation on other open-source platforms similar to SMIP and improving the data rates we have achieved. Moreover, we in early stages of adding functionalities of closed loop feedback system which can enhance the SM implementation to go beyond only monitoring. The closing of the loop aspect will allow for real-time update of the parameters in case of any deteriorating or undesired performance in the metrics. Furthermore, with the growing adoption of 5G technologies, we intend to scale our SM implementation to make it compatible for a network of connected machines that allow data collection, analyses, and communication channel with lower latencies, higher reliability, and bandwidth.

Acknowledgements

The authors gratefully acknowledge the generous support from Texas A&M University X-Grants and Rockwell International Professorship. They would also like to thank CESMII, especially Mr. Jonathan Wise, VP and Chief

Technology Architect, for his valuable feedbacks throughout the project.

References

[1] Perez, A. T. E., Rossit, D. A., Tohme, F., & Vásquez, Ó. C. (2022). Mass customized/personalized manufacturing in Industry 4.0 and blockchain: Research challenges, main problems, and the design of an information architecture. *Information Fusion*, *79*, 44-57.

[2] Botcha, B., Wang, Z., Rajan, S., Gautam, N., Bukkapatnam, S. T., Manthanwar, A., ... & Korambath, P. (2018, June). Implementing the transformation of discrete part manufacturing systems into smart manufacturing platforms. In *International Manufacturing Science and Engineering Conference* (Vol. 51371, p. V003T02A009). American Society of Mechanical Engineers.

[3] Jones, M. D., Hutcheson, S., & Camba, J. D. (2021). Past, present, and future barriers to digital transformation in manufacturing: A review. *Journal of Manufacturing Systems*, *60*, 936-948.

[4] Iquebal, A. S., Wang, Z., Ko, W. H., Wang, Z., Kumar, P. R., Srinivasa, A., & Bukkapatnam, S. T. (2018). Towards realizing cybermanufacturing kiosks: Quality assurance challenges and opportunities. *Procedia Manufacturing*, *26*, 1296-1306.

[5] Tiwari, A., Reddy, A. N., & Bukkapatnam, S. T. (2020). Cybersecurity assurance in the emerging manufacturing-as-a-service (MaaS) paradigm: A lesson from the video streaming industry.

[6] Modelski, G., & Perry III, G. (2002). "Democratization in long perspective" revisited. Technological Forecasting and Social Change, 69(4), 359-376.

[7] Niehaves, B. (2011, January). Democratizing process innovation: a comparative study of public sector business process management networks. In 2011 44th Hawaii International Conference on System Sciences (pp. 1-9). IEEE.

[8] Forbes, H., & Schaefer, D. (2017). Social product development: the democratization of design, manufacture and innovation. Procedia Cirp, 60, 404-409.

[9] Kurfess, T. R., Saldana, C., Saleeby, K., & Dezfouli, M. P. (2020). A review of modern communication technologies for digital manufacturing processes in industry 4.0. *Journal of Manufacturing Science and Engineering*, 142(11).

[10] Kurfess, T., Saldana, C., Saleeby, K., & Parto-Dezfouli, M. (2020). Industry 4.0 and Intelligent Manufacturing Processes: A Review of Modern Sensing Technologies. *Journal of Manufacturing Science and Engineering*, 142(11).

[11] Kusiak, A. (2019). Service manufacturing: Basic concepts and technologies. Journal of Manufacturing Systems, 52, 198-204.

[12] Harris, G., Yarbrough, A., Abernathy, D., & Peters, C. (2019). Manufacturing readiness for digital manufacturing. *Manufacturing Letters*, *22*, 16-18.

[13] Antomarioni, S., Lucantoni, L., Ciarapica, F. E., & Bevilacqua, M. (2021). Data-driven decision support system for managing item allocation in an ASRS: A framework development and a case study. *Expert Systems with Applications*, 185, 115622.

[14] Lu, Y., Xu, X., & Wang, L. (2020). Smart manufacturing process and system automation–a critical review of the standards and envisioned scenarios. *Journal of Manufacturing Systems*, *56*, 312-325.

[15] Kang, H. S., Lee, J. Y., Choi, S., Kim, H., Park, J. H., Son, J. Y., ... & Noh, S. D. (2016). Smart manufacturing: Past research, present findings, and future directions. *International journal of precision engineering and manufacturing-green technology*, *3*(1), 111-128.

[16] Woodward Steve, Attachments – Budgetary proposal and system overview, April 2016.

[17] Clean Energy Smart Manufacturing Innovation Institute (CESMII), Office of Energy Efficiency & Renewable Energy, U.S. Department of Energy. https://www.cesmii.org.

[18] Alqoud, A., Schaefer, D., & Milisavljevic Syed, J. (2021). Industry 4.0: Retrofitting of Legacy Machines for Smart Manufacturing.

[19] Meesublak, K., & Klinsukont, T. (2020, August). A cyber-physical system approach for predictive maintenance. In *2020 IEEE International Conference on Smart Internet of Things (SmartIoT)* (pp. 337-341). IEEE.

[20] Strauß, P., Schmitz, M., Wöstmann, R., & Deuse, J. (2018, December). Enabling of predictive maintenance in the brownfield through low-cost sensors, an IIoT-architecture and machine learning. In *2018 IEEE International conference on big data (big data)* (pp. 1474-1483). IEEE.

[21] Verma, A., Goyal, A., Kumara, S., & Kurfess, T. (2021). Edge-cloud computing performance benchmarking for IoT based machinery vibration monitoring. *Manufacturing Letters*, *27*, 39-41.

[22] The NI TDMS file format, National Instruments, updated June 2021, https://www.ni.com/en-us/support/documentation/supplemental/06/the-ni-tdms-file-format.html.

[23] What is the GraphQL Foundation? 2021. https://graphql.org/foundation/

[24] Botcha, B., Rajagopal, V., & Bukkapatnam, S. T. (2018). Process-machine interactions and a multi-sensor fusion approach to predict surface roughness in cylindrical plunge grinding process. *Procedia Manufacturing*, *26*, 700-711.

[25] Botcha, B., Iquebal, A. S., & Bukkapatnam, S. T. (2020). Smart manufacturing multiplex. *Manufacturing Letters*, *25*, 102-106.

[26] Mahesh, P., Tiwari, A., Jin, C., Kumar, P. R., Reddy, A. N., Bukkapatanam, S. T., ... & Karri, R. (2020). A survey of cybersecurity of digital manufacturing. *Proceedings of the IEEE*, *109*(4), 495-516.