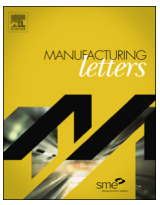
Available online at www.sciencedirect.com

Manufacturing Letters

Manufacturing Letters 44 (2025) 1349–1356



53rd SME North American Manufacturing Research Conference (NAMRC 53, 2025)

Automotive Industrial Non-scalar Data Sharing

Debejyo Chakraborty[a,*], Bernie Gallis[a], Jerome Schroeder[b], Paul Wright[c], Michael King[d]

[a]*General Motors, Warren, MI, 48092, USA*
[b]*General Motors (retd.), Warren, MI, 48092, USA*
[c]*Allegion, Indianapolis, IN, USA*
[d]*LHP Solutions, USA*

Abstract

Industry 4.0 was coined over a decade ago, outlining the concept of connected devices on the plant floor. Since then the industry has been busy establishing connectivity without any guidance. Each company adopted the method of their choice and the architecture was motivated by economics. Today, we are at a time where this unguided behavior has made it difficult to access data from our own processes and integrate equipment from multiple suppliers to achieve a task. A United States Council for Automotive Research (USCAR) standard was published to mitigate this impediment. In this document we have furthered that standard by proposing a standard data file architecture and a sharing mechanism from plant floor.

Keywords: data sharing; file format; industry 4.0; binary file; sequential data;

1. Introduction

Automotive companies would remain competitive in the era of smart manufacturing [1] if data connectivity and integration between systems utilized open standards. Supporting legacy, proprietary closed systems on the manufacturing floor has created an unacceptable burden of lost time, talents, and profit. The makers of industrial control equipment currently in use by the United States (US) automakers fall along a continuum of readiness to adopt open standards. To understand the current commercial market practice, twenty original equipment manufacturers (OEMs) were surveyed by us as a part of a United States Council for Automotive Research (USCAR) initiative. Among them, eleven were entities that comprised standards organizations. Additionally, there were companies from other relevant industries such as automation controllers, process controllers, sensors and hubs, network devices, other practitioners, integrators, and several industry-recognized subject matter experts. They were all benchmarked as inputs to understand the gap and the cause in data communication standards. Some of the largest makers of automation controllers actively resist the adoption of open standards as it threatens their longstanding revenue model. Conversely, a market of process controller companies are aggressively pursuing open standards and innovative product strategies to meet the requirements of smart manufacturing. Throughout this landscape two primary communication protocols dominate the marketplace: open platform communications unified architecture (OPC-UA) [2], and message queuing telemetry transport (MQTT) [3]. The OPC-UA was favored by legacy automation controllers whereas MQTT was featured by innovative process controllers. A number of integrators and systems experts advised us that both these protocols would exist for the foreseeable future. Therefore, efforts to standardize should be focused at the data syntax level, not the communication protocol.

Manufacturing devices present data to information technology (IT) applications in varied and often proprietary methods, as defined by the equipment vendors. Data communication from

* Corresponding author.
E-mail address: debejyo.chakraborty@gm.com (Debejyo Chakraborty).

these devices is usually developed from scratch for each new device type or vendor. Here are a few examples.

- Matrox cameras retrieve image and text measurement files via file transfer protocol (FTP) [4, 5] or server message block (SMB) [6, 7].
- Atlas Copco Dispense communicates using internal web server serving extensible markup language (XML) via hypertext transfer protocol (HTTP) [6, 7].
- Nordson Dispense utilizes programmable logic controller (PLC) based communication.
- WTC Weld provides data to our custom representational state transfer (REST) [6, 7] endpoint.
- Rockwell PLCs use Kepware [8], OPC-UA and polling.
- Fanuc Robots utilize a closed system Zero Down Time (ZDT) [9], and cloud-based Software as a Service (SaaS).
- Banner Vibration communicates using complex configuration of devices over Modbus [10] to Kepware.
- KCF Vibration is closed smart diagnostics system that cloud-based utilizing SaaS.
- Rockwell Vibration uses ControlLogix® [11] based approach passing data through a PLC.

Some of the methods could be inherently insecure due to the age or limited processing capabilities of the devices. Some methods are intentionally closed to enable vendors to sell additional software products or "analytics services." As a result, the customer does not get to access their own data without adopting a non-efficient and circuitous process, or paying additional development or licensing fees. Consequently, obtaining data from plant floor has become extremely challenging and expensive.

To alleviate this pain point, it was essential to develop a specification for operational technology (OT) equipment data publishing, and include it with equipment purchasing specifications. The intent for this non-rigid specification was to allow all new equipment to easily plug into device level analytics (DLA) data collection systems. Specifications of this nature could include security requirements and external/third party data sharing boundaries.

Motivated by the needs of the industry, USCAR published a data communication standard [12]. The standard preferred MQTT as a communication format and outline plant floor events that would trigger such communication. It also provided the structure for data payload and the minimum content requirement for the payload. In the current version, the standard did not have a prescriptive outline for data nomenclature [13]. While that might be a paramount task, it might become essential if we do not want to maintain custom "dictionary" for each vendor and each equipment. In addition, the standard did not prescribe a standard file format to store and communicate multidimensional sequential data [14], also commonly known as waveform or time-series when the domain is time.

The focus of this document has been to prescribe a file format to store and communicate multidimensional sequential data. It also recommends a general organization of the such data in a manner that allows the most commonly used modern programming languages to read, write and share the data.

The remainder of this document is organized as follows. Section 2 discusses types of industrial data at a high level. This classification of data was motivated by the tasks our industry needs to perform. It segued into data file format proposal in Section 3. A possibly simplistic data sharing mechanism was then discussed in Section 4. Section 5 illustrated our first application that motivated the use of the suggested file format and sharing mechanism. Concluding remarks and the recommend next steps are provided in Section 6.

Nomenclature

ASCII	American standard code for information interchange
CSV	comma separated value
DLA	device level analytics
FTP	file transfer protocol
HDF5	hierarchical data format version 5
HTTP	hypertext transfer protocol
IT	information technology
MQTT	message queuing telemetry transport
OEM	original equipment manufacturer
OPC-UA	open platform communications unified architecture
OT	operational technology
PLC	programmable logic controller
PMQ	process monitoring for quality
REST	representational state transfer
SaaS	Software as a Service
SMB	server message block
US	United States
USCAR	United States Council for Automotive Research
UTC	universal time coordinated
XML	extensible markup language
ZDT	Zero Down Time

2. Industrial Data Classification

Data architecture and communication within a smart manufacturing environment allows for many different types of data along with different ways in which data is communicated between machines on the manufacturing floor. We introduced a broad data classification in [12], that is re-illustrated in Figure 1 to aid the discussion that follows.

Industrial data are of two kinds, *job related* and *non-job related*. "Job" here is defined as the act of making a part in production. Production of a part involves the part itself and the machine that acts on the part. A "machine" is defined as the entity that is capable of either modifying a part or measuring an attribute of it. Naturally, job related data can be categorized into *part* and *machine* data.

A part data consists of the part *identifier*, *specification*, and *quality* information. The part identifier is a unique entity that

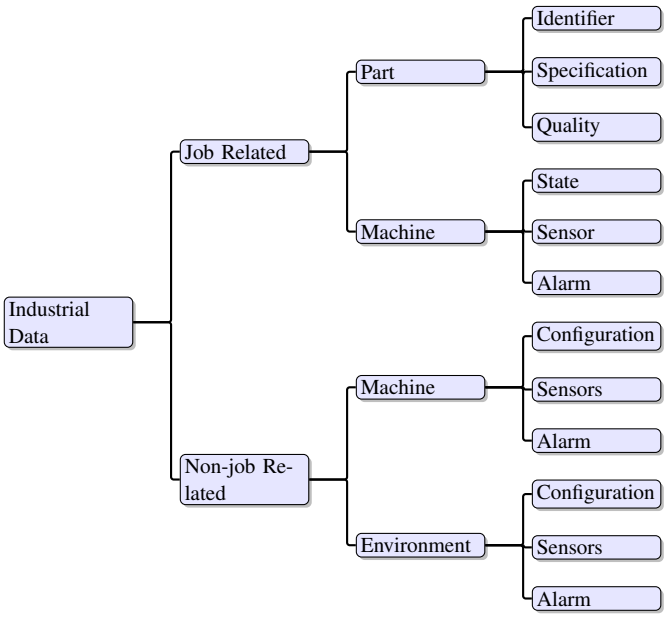


Figure 1. Classification of industrial data.

identifies a part, for example, a serial number. The specification data pertains to the manufacturing requirements of the part, for example, dimensions and tolerances. It may also consist of physical and functional attributes of the part. After an action is taken on a part, it is expected to pass certain quality checks. The outcome of these measurements may be as simple as a Boolean or something very descriptive and multi-dimensional. The data of this outcome may include the measurement along with the decision.

A machine may or may not be directly related to a job. The type of data reported is different in each category. Machine data is job related when the machine is actively making, altering, or measuring a part. It can include machine *state*, data from the *sensor* attached to the machine, or an *alarm*. The state communicates the operating mode that the machine is in, with "idle" as a valid state and should technically indicate that the associated data is not job related. When a machine is not idle, it is expected to record data from its sensors. These data may be (seldom) scalars, time-series signals, images, or video. When data is scalar, it could be communicated as a part of MQTT payload and fetch a place in a database. However, when data is large and multidimensional, it may be too big for a MQTT payload to communicate effectively and would surely require some file format to package into for storage as indicated in Section 3.

One may argue that time-series databases [15] address the standard packaging of data. However, the implementation and usage to develop analytics on those databases require specific skillset and may not be appealing without a large IT infrastructure and subsequent maintenance budget, especially when we consider high frequency multidimensional time series data like a high speed video or multiple high sampling rate sensors. A file, on the other hand, provides the needed simplicity and agility. More often than not, time series data from a job is consumed as a whole and querying specific points in the time series is seldom useful. Time series data analysis involve reading a whole sequence, transforming and decimating it to extract meaningful features that are indicative of either the job or the part. If reading a subset of time series information is desired, the file formats proposed here enable such access.

Unlike sensor data, an *alarm* from a machine is usually thought of as a Boolean. However, additional information is necessary to describe the alarm. An alarm is triggered if the data from a sensor violates some threshold set by the configuration of the machine. So, the sensor on which the alarm was triggered and the time-stamp at which such an event took place must be packaged with this data. Such time-stamped information is instrumental in correlating other associated events.

Non-job related machine data is obtained when the machine is in "idle" state. So, recording the state is redundant here. However, this is the opportunity to communicate the *configuration*(s) of this machine. It could contain a library of operating configurations and the conditions that require any of those. An idle machine also has information to provide. The *sensor*s in a machine could be active and reporting data periodically to keep track of any condition of interest. Such information can help in tearing readings in non-idle state. Machine calibration could also be one of those non-job related sensor data opportunities. Machine *alarm*s in this state would generally be indicative of maintenance and tool/sensor change.

The *Plant/environment* data is essential to keep track of the overall conditions for manufacturing. For example in a dry-room, humidity and temperature are essential to be monitored at all times. Naturally, we need to know what the environmental requirements are for the given manufacturing process. Such information is captured in the *configuration* data for the plant. *Sensor*s then become an obvious necessity and data from it would need to be communicated on a predefined format and interval. Any violations of desired configuration, as indicated by any sensor, would then need to be recorded as an *alarm*. Such an alarm should at least contain sensor information and the timestamp of the event.

Non-job related machine or environment data could also be time-series. Unlike job-related machine data, non-job related time-series data is unbound in time and generally collected in long intervals. Such data are excellent candidates for time-series databases. The usage of these time-series data are usually to monitor long-term trends, correlation of events with alarms and so on, that require more of selective querying than complex mathematical transforms.

3. Standard Data File Formats

Storing non-scalar data has been an ongoing challenge. In manufacturing non-scalar data typically involves image [16], video [17] and multidimensional sequential data. The most versatile format for storing data would be as a general American standard code for information interchange (ASCII) file [18]. In such a file each digit of a number, a symbol or a letter takes up the same amount of space, 1 byte. Plain text files are examples of such files. Manufacturing data could obviously be arranged

with special characters such as tab, comma, semicolor, etc. as delimiters. Depending on the delimiter used, the file could be referred to as a comma separated value (csv), tab separated value (tsv) and so on. In fact, a csv file came into use as early as 1972 and was first supported by the IBM Fortran compiler under OS/360. Storing numerical data in this manner is universally readable, but results in a very large file. For example, storing a double precision number takes only 8 bytes of memory in a binary file. To represent the number in an ASCII file it would take 19 bytes (17 digits, one decimal point and one sign). To practically asses this size difference the code in the Appendix could be used. The comparison would should show that the ASCII file is about 2.3 times larger than an uncompressed binary file when storing exactly the same data.

Data in binary files could be stored as compressed or uncompressed [19, 20]. In our example, we used an uncompressed file to make it a fair comparison. Compression benefits can vary depending on the kind of compression and the content of the file. A file containing no pattern, and essentially random numbers like the one we used in the provided code would yield the lowest compression ratio. To improve compression ratio, some data accuracy could be sacrificed by using lossy compression techniques. The larger the loss, the better the compression. Table 1 lists the most commonly used image file formats and indicates which ones are compressed and lossy. Depending on the use of the images, some level of loss in data could be sustained, making lossy compression a viable option. Similar is the case with video files except that the video files are a bit more complex than image files. Video files contain a sequence of images as well as audio stream. Table 2 lists the commonly used video file formats.

Table 1. Common image file formats.

Image Format	Compression	Lossy
BMP		
RAW (CR2, NEF, ARW, DNG)		
TIFF (Uncompressed)		
TIFF (Lossless)	✓	
PNG	✓	
GIF	✓	
FLIF	✓	
JPEG 2000 (Lossless)	✓	
WebP (Lossless)	✓	
JPEG	✓	✓
JPEG 2000 (Lossy)	✓	✓
WebP (Lossy)	✓	✓
HEIF / HEIC	✓	✓

It would be natural to consider audio files for storing sequential data, especially time-series data. There are widely accepted audio file standards as well, as shown in Figure 3. It would be a pertinent choice, but with limitations. Audio file domain is regularly sampled in time and can only contain a limited number of sequences, each corresponding to an audio channel. There are limited and predefined metadata fields for these file types. For these reasons, an audio file format is not suitable for a multidimensional sequential data.

Table 2. Common video file formats.

Video Format	Compression	Lossy
YUV (raw)		
AVI		
HuffYUV	✓	
FFV1	✓	
Apple ProRes (lossless)	✓	
Apple ProRes (lossy)	✓	✓
MP4 (H.264, H.265, AV1)	✓	✓
MPEG-2	✓	✓
VP9	✓	✓
WMV	✓	✓

Table 3. Common audio file formats.

Audio Format	Compression	Lossy
WAV		
AIFF		
PCM (raw)		
FLAC	✓	
ALAC	✓	
WMA (lossless)	✓	
WMA (lossy)	✓	✓
MP3	✓	✓
AAC	✓	✓
OGG Vorbis	✓	✓
Opus	✓	✓

When it comes to files for storing multidimensional sequential data, there has not been any universal consensus. Vendors often use proprietary formats to package data in their application in the pretext of information security, locking the customer out of its own data. There are some commonly used established file formats that are widely understood and binary, as shown in Table 4. The application of these files generally prohibit lossy compression and has been consequently omitted in this context. It is clear that MAT file format is the oldest binary file format. Though it was introduced by Mathworks for storing Matlab® workspace variables, the format has been openly communicated making it suitable for adoption by a wide variety of scientific programming languages. A close popular contender was introduced over a decade later, hierarchical data format HDF (with the more recent version 5, HDF5®), for storing and organizing large scientific datasets. Both of these formats appear in the

Table 4. File type options for multidimensional sequential data.

File Format	First Introduced	Binary Format
CSV [21]	1972	
MAT [22]	1984	✓
NetCDF [23]	1989	✓
XML [24]	1996	
HDF5® [25, 26]	1998	✓
JSON [27]	2001	
Parquet [28]	2013	✓

Library of Congress Dataset formats description [29] as recommended binary file formats. In fact, the latest MAT file is based on HDF5® and supports very large data files unlike its predecessors. These file formats have been openly documented and have been adopted by various scientific programming languages including Python, R, Wolfram Mathematica, Matlab®, GNU Octave, C and C++. These formats allow us to read in a portion of the data file and not have to load the entire file into memory for reading. The ease and speed of reading/writing MAT file from the mentioned programming languages, especially Matlab® has motivated us to gravitate towards MAT file as our standard. MAT file is also openly documented and supported by Mathworks. HDF5® is very generic and comes with overheads which could easily be eliminated by choosing the appropriate MAT file version when very large files (larger than 2GB) are not required.

Most recently Parquet was introduced for columnar storage format for big data analytics. Data stored in this format is not as flexible as MAT or HDF5®. Parequet requires data in all columns to be of the same length. So, sequential data with different lengths cannot be stored in Parquet. In addition, Parquet has been tested to be among the slowest in read/write when used with Matlab®.

Data without metadata is incomplete. So, any file should have a suggested framework for storing all relevant metadata along with the data for completeness and meaningful interpretation. MAT and HDF5® files allow us to store metadata along with the data. Having used MAT file to store data from a wide variety of manufacturing processes, we have converged on the following scheme.

The file primarily contains four predefined variables

- `metadata`
- `header`
- `data`
- `units`

with an optional addition of at least two, such as

- `strheader`
- `strdata`.

All columns in `data` is required to contain the same number of datapoints. In applications where that is not true or data does not share a common sampling domain, additional variables sets as `data1`, `data2`, etc.. along with their respective `header1`, `units1`, `header2`, `units2` and so on could be used. It is conventional for us to use the first column of the `data` variable to be the domain. In time-series data, it would be time values. However, in regularly sampled time series data, one could skip that entirely and note the sampling rate as a part of metadata to further optimize file size.

All scalar metadata information is stored in `metadata` as a two column list, known as *cellarray* in Matlab®. The first column contains the key and the second column contains the value. For example, `metadata` can be arranged as follows.

'Serial Number'	'S12345'
'Timestamp'	'2020-12-31T23:59:59.999Z'
'Global count'	500
'Operation'	'1'
'Plant Code'	'5'

Note that the timestamp follows a format specified in ISO 8601 (which also would allow '20201231T235959.999Z') and is always reported as Universal Time Coordinated (UTC). The metadata typically contains a part identifier that is being manufactured such as a timestamp, information about the operation, and the plant. Information about the machine performing the job could also be included. Though metadata information generally exists in a database to which such files could be linked [30], it is a commendable practice to populate the metadata with all the necessary information. If the file gets orphaned from its database the metadata in the file would provide essential context.

The `header` is a list (or a cellarray) of labels for the various data arrays. For time series data, the first entry would invariably be "Time." The `data` contains data organized in columns. For example, let us consider a machine reporting voltage, current and temperature data. The `data` would contain four columns, "Time," "Voltage," "Current" and "Temperature." The `header` would contain those strings in the same order in which the columns in `data` appears. Obviously in this example the `units` will contain a list {'second', 'volt', 'ampere', 'centigrade'} denoting the appropriate units.

There could be uncommon scenarios when an array of data reported is not numeric. The `strdata` and `strheader` pair could be used for that purpose. Consider a domain of discrete labels. The labels could be stored in `strdata` with the appropriate `strheader`. An example of this is values reported from a gas chromatography system. The calculated numeric values stored in `data` would correspond to names of calibrated gases stored in `strdata`.

A MAT or a HDF5® file could also be used to store video, image, audio data and any other data. More than two dimensions (row and column) are supported in these file formats.

4. Data Sharing

Data generated in complex manufacturing processes could be bountiful. While that enables big-data-big-model [31] kind of analytics, a new challenge of transferring such data from the edge device, and subsequently sharing it for analysis, is encountered. The type of data dictates the communication mechanism. Metadata and scalar information, or even sequential data can be shared through any *publish-subscribe* messaging pattern [32, 33], but the same could be challenging or impossible for large sequential data and data files containing high resolution images, videos or other multi-dimensional data. As the standard [12] suggests, it would be best to package such information in appropriate file formats indicated in Section 3, and the link shared as a payload content of a message in the publish-subscribe pattern like MQTT.

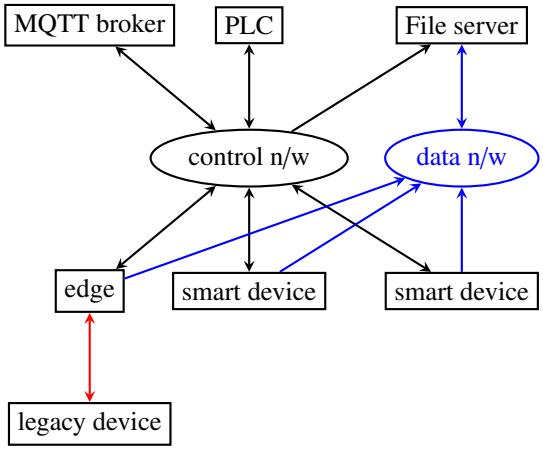


Figure 2. Data flow concept. Network is abbreviated as n/w for brevity.

An edge device now would require not only publish-subscribe capabilities, but also file transfer abilities. The onus of perpetually hosting such files should never be on the edge device. Therefore, a connection to a file server must also exist. Depending on the frequency and size of such data files, the primary control network to which these devices are connected, may not have enough bandwidth. It might be a smart design structure to provide a secondary physical layer for data transfer, which may also be expensive if the required bandwidth exceeded 1 Gbps.

Consider an ideal dataflow scenario as shown in Figure 2. There are two networks, a control network and a data network. It is commonplace to have tiered network on a plant floor [34]. A control network could be EtherCAT [35], or there could be a sub-network with EtherCAT dedicated for real-time control traffic. The control network is used for limited traffic like control signals and MQTT publishing. The data network is a more expensive high bandwidth network that is dedicated for transporting large data files. In modern plant floor systems with high resolution data, it might even be wise to have a network that support an excess of 10 Gbps.

Smart devices [36] can connect to both networks and manage traffic appropriately. Legacy devices can also be integrated in this ecosystem through an edge computer, making it a "smart device". Another edge device on the control network could act as the MQTT broker. PLC device is also connected to the control network. All the smart devices could be connected to the file server through the data network. Since the data network could be a part of a larger enterprise level ecosystem, the edge devices are not expected to receive incoming traffic from such a network.

Transferring of the data file(s) could be achieved by a variety of mechanisms, including a primitive socket based communication. However, there are standard network protocols to achieve it, as listed in Table 5. Comparing the throughput and latency, SMB would be the most preferred, though not the most secure. In a properly firewalled intranet, SMB it is the most user-friendly and secure protocol. When encrypted transfer over broader internet is required, SFTP provides most secure option.

Table 5. Comparison of Standard & Network-Based File Transfer Protocols where L stands for latency and TP stands for throughput.

Protocol	Security	L (ms)	TP (MB/s)
FTP	None	30–50	40–80
FTPS	TLS	30–50	40–80
SFTP	SSH-based	40–70	10–30
TFTP	None	10–20	5–10
NFS	Optional (Kerberos)	1–10	50–120
SMB	NTLM, Kerberos	5–20	40–100
HTTPS	TLS	50–100	5–50

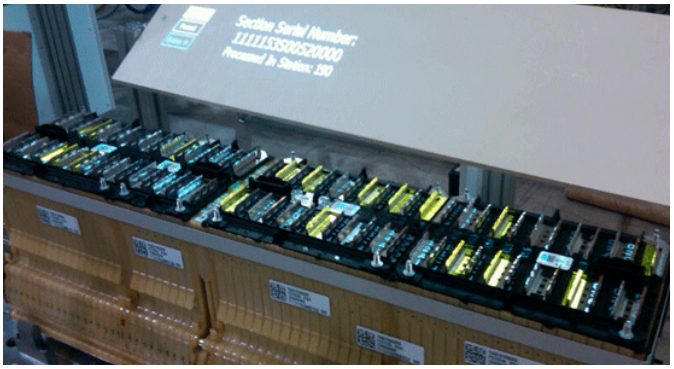


Figure 3. A module of a Chevrolet Volt Gen I battery, with the suspect welds highlighted with yellow light [38, 39].

5. Application

The first application of this concept was in 2011. GM was making battery packs for Chevrolet Volt Generation I in Brownstown Battery Assembly plant [37]. A battery module is shown in Figure 3. In the battery modules, lithium ion cell tabs were being ultrasonically welded to the bus bars to form a series-parallel connection. These welds did not only have to make a sound electrical connection, but also had to hold up mechanically to expected driving conditions. While the electrical connection could be verified directly, there was no way to ascertain the mechanical integrity of these welds non-destructively. An industry-first process monitoring for quality (PMQ) method was invented [31] and implemented [40] to ascertain the quality 100% of these welds before it left the factory floor. The PMQ system utilized high resolution (sampled at 100 kHz) signals from the welding process to predict the quality of the weld. These signals were stored as a .mat file in a scheme that is closely aligned with the one described in Section 4. Since our machine learning experts were from different background and chose various programming languages, the file format had to be chosen that could be easily read and parsed by all these languages. We started with a CSV file just to realize how big those were and how slow they were to read and write.

The network connection topology was also similar to Figure 2. The data acquisition computer ("edge") had access to the PLC network and the plant floor secondary network. The PLC network was used to communicate with the weld controller and the station controller. Signals were acquired directly from ana-

log sensors. The information from the two sources were packaged and saved locally. Decision on the good/suspect welds were communicated back to the controller over the PLC network. A secondary "file server" collected these .mat files and organized the information in a database.

This large repository of files were then periodically physically transported to Research and Development using a mechanism similar to "sneakernet" [41, 42]. In fact this is a data transfer practice by large data firms to circumvent slow network bandwidth [43, 44, 45]. These data files were then utilized by a remote high performance computers to retrain new machine learning models, that got redeployed in the plant.

6. Conclusion

In the evolution of industry 4.0 we are at point in time when we need to dedicate resources in standardizing data communication. Without a standard equipment manufacturers are developing their own custom ecosystem, making it very expensive to integrate equipment from different vendors. Moreover, the data generated by these equipment are often stored in proprietary binary format and it prevents the customer from utilizing it for analytics without bearing additional expenses. A standard by USCAR, USCAR-53, has been published to address standard data communication protocol and strategy with a definition of the minimum payload. Some aspects that remained unsaid in the current version of the standard has been discussed in this document. A standard data file format for storing general multidimensional sequential data has been proposed through a preliminary comparison of merits and demerits of commonly used file formats. A network architecture for plant floor systems has also been proposed, with a suggestion of communication protocol, that enables communication of large data files. A practical example of the first use of a similar ecosystem was described that was the precursor for developing these standards. While there may never be the perfect solution, it should not hinder us from converging on a consensus regarding data file standards based on the past few decades of experience. A standard data would benefit all in the industry. The suppliers would not have to invent, maintain, and support a standard of their own. The original equipment manufacturers would not have to dedicate immense IT resources to write custom data parsers and interfaces for every equipment and create a very complex diverse ecosystem of data and control. The next steps that remain to be taken would be to define common data model and database architecture for storing common widely used manufacturing processes. The expectation is that eventually we can converge on one system that works.

Acknowledgment

The authors would like to thank the Manufacturing Technical Leadership Council members and the Smart Manufacturing Workgroup participants of USCAR for their support in establishing the standard for data communication.

References

[1] K. Haricha, A. Khiat, Y. Issaoui, A. Bahnasse, H. Ouajji, Recent technological progress to empower smart manufacturing: Review and potential guidelines, IEEE Access 11 (2023) 77929–77951. doi:10.1109/ACCESS.2023.3246029.

[2] Open platform communications unified architecture, Online. URL https://opcfoundation.org/about/opc-technologies/opc-ua/

[3] Message queuing telemetry transport, Online. URL https://mqtt.org/

[4] J. Postel, J. Reynolds, File transfer protocol, Tech. Rep. 959, RFC Editor (1985). doi:10.17487/RFC0959. URL https://www.rfc-editor.org/info/rfc959

[5] S. P. Singh, N. Goyal, Security configuration and performance analysis of file transfer protocol, International Journal of Computer and Communication Technology 2 (2) (2014) 106 – 109. URL https://ijccts.org/index.php/pub/article/view/195

[6] A. S. Tanenbaum, D. J. Wetherall, Computer Networks, 5th Edition, Prentice Hall, 2010.

[7] W. Stallings, Data and Computer Communications, 10th Edition, Prentice Hall, 2013.

[8] Guide — secure kepware server deployment, Online (Jan. 2023). URL https://community.ptc.com/sejnu66972/attachments/sejnu66972/kepware/1609/1/Secure%20Kepware%20Server%20Deployment.pdf

[9] FANUC, Zero down time (zdt), Online. URL https://www.fanucamerica.com/products/robots/zdt-zero-down-time

[10] Modbus protocol tutorial : Frame formats for rtu, ascii and tcp communication, Online. URL https://www.rfwireless-world.com/Tutorials/Modbus-Protocol-tutorial.html

[11] Controllogix control systems, Online. URL https://www.rockwellautomation.com/en-us/products/hardware/allen-bradley/programmable-controllers/large-controllers/controllogix.html

[12] USCAR/SAE, IIoT data communication standard, Tech. Rep. SAE/USCAR-53, SAE International (May 2023).

[13] Sustainable manufacturing – cesmii and plattform industrie 4.0, Online (2021). URL https://www.cesmii.org/sustainable-manufacturing-cesmii-and-plattform-industrie-4-0/

[14] F. J. Király, H. Oberhauser, Kernels for sequentially ordered data, Journal of Machine Learning Research 20 (31) (2019) 1 – 45. URL http://jmlr.org/papers/v20/16-314.html

[15] S. K. Jensen, T. B. Pedersen, C. Thomsen, Time series management systems: A survey, IEEE Transactions on Knowledge and Data Engineering 29 (11) (2017) 2581–2600. doi:10.1109/TKDE.2017.2740932.

[16] R. C. Gonzalez, R. E. Woods, Digital Image Processing, 3rd Edition, Pearson, 2007.

[17] J. W. Woods, Multidimensional Signal, Image, and Video Processing and Coding, 2nd Edition, Academic Press, 2011.

[18] C. Mackenzie, Coded Character Sets, History and Development, Addison-Wesley Publishing Company, 1980. URL https://textfiles.meulie.net/bitsaved/Books/Mackenzie_CodedCharSets.pdf

[19] K. Sayood, Introduction to data compression. a volume in the morgan kaufmann series in multimedia information and systems, Elsevier (2006) 703.

[20] S. Shirani, Data compression: The complete reference (by d. salomon; 2007)[book review], IEEE Signal Processing Magazine 25 (2) (2008) 147–149.

[21] Y. Shafranovich, Common format and mime type for comma-separated values (csv) files, RFC Editor (4180). URL https://www.rfc-editor.org/rfc/rfc4180

[22] I. The MathWorks, Mat-file format. URL https://www.mathworks.com/help/pdf_doc/matlab/

matfile_format.pdf
[23] Unidata, netCDF User's Guide (2023).
URL https://docs.unidata.ucar.edu/netcdf/latest/user/index.html
[24] T. Reese, The extensible markup language (xml), Information Technology and Libraries 16 (2) (1997) 78–81.
[25] HDF5 support.
URL https://portal.hdfgroup.org/display/HDF5/HDF5
[26] M. Folk, G. Heber, An overview of the hdf5 technology suite and its applications, Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases (2011) 36–47.
[27] D. Crockford, Introducing json (2001).
URL https://www.json.org/json-en.html
[28] D. Vohra, Apache parquet, Pro Hadoop (2016) 325–335.
[29] Format descriptions for dataset formats.
URL https://www.loc.gov/preservation/digital/formats/fdd/dataset_fdd.shtml
[30] D. Chakraborty, M. McGovern, NDE 4.0: Smart NDE, in: 2019 IEEE International Conference on Prognostics and Health Management, 2019.
[31] J. A. Abell, D. Chakraborty, C. A. Escobar, K. H. Im, D. M. Wegner, M. A. Wincek, Big data driven manufacturing — process-monitoring-for-quality philosophy, Journal of Manufacturing Science and Engineering 139 (10) (2017) 101009–1 – 101009–12. doi:10.1115/1.4036833.
[32] H. Raddatz, E. Mahmoud, F. Hölzke, P. Danielis, D. Timmermann, F. Golatowski, Evaluation and extension of opc ua publish/subscribe mqtt binding, in: 2020 IEEE Conference on Industrial Cyberphysical Systems (ICPS), Vol. 1, 2020, pp. 543–548. doi:10.1109/ICPS48405.2020.9274696.
[33] R. Joshi, P. Didier, J. Jimenez, T. Carey, The industrial internet of things volume g5: Connectivity framework, Tech. rep., Industrial Internet Consortium (2018).
[34] F. Treacy, Accelerating the transition to industry 4.0 with industrial ethernet connectivity, Tech. rep., Analog Devices (2020).
URL https://www.analog.com/en/thought-leadership/acceler-trans-to-ind-4-pt-0-with-ind-ethernet-connect.html
[35] K. Langlois, T. van der Hoeven, D. Rodriguez Cianca, T. Verstraten, T. Bacek, B. Convens, C. Rodriguez-Guerrero, V. Grosu, D. Lefeber, B. Vanderborght, EtherCAT tutorial: An introduction for real-time hardware communication on windows, IEEE Robotics & Automation Magazine 25 (1) (2018) 22–122. doi:10.1109/MRA.2017.2787224.
[36] S. Poslad, Ubiquitous Computing: SmartDevices, Environments and Interactions, Wiley, 2009.
[37] E. C. Report, Gm using ultrasonic welding in the cadillac elr battery pack, accessed: 2025-02-16 (2013).
URL https://electriccarsreport.com/2013/08/gm-using-ultrasonic-welding-in-the-cadillac-elr-battery-pack/
[38] J. P. Spicer, J. A. Abell, M. A. Wincek, D. Chakraborty, J. Bracey, H. Wang, P. W. Tavora, J. S. Davis, D. C. Hutchinson, R. L. Reardon, S. Utz, Quality status display for a vibration welding process, United States Patent and Trademark Office (US9827634B2).
URL https://ppubs.uspto.gov/dirsearch-public/print/downloadPdf/9827634
[39] J. P. Spicer, J. A. Abell, M. A. Wincek, D. Chakraborty, J. Bracey, H. Wang, P. W. Tavora, J. S. Davis, D. C. Hutchinson, R. L. Reardon, S. Utz, Quality status display for a vibration welding process, United States Patent and Trademark Office (US9604305B2).
URL https://ppubs.uspto.gov/dirsearch-public/print/downloadPdf/9604305
[40] J. P. Spicer, D. Chakraborty, M. Wincek, J. Abell, Implementation strategy for launch and performance improvement of high throughput manufacturing inspection systems, in: Proceedings of the 52nd SME North American Manufacturing Research Conference (NAMRC) published in Manufacturing Letters, Vol. 41, 2024, pp. 143–152. doi:https://doi.org/10.1016/j.mfglet.2024.09.018.
[41] A. C. Jaya, C. Safitri, R. Mandala, Sneakernet: A technological overview and improvement, in: 2020 IEEE International Conference on Sustainable Engineering and Creative Computing (ICSECC), 2020.
URL https://ieeexplore.ieee.org/iel7/9557351/9557355/09557509.pdf
[42] J. Gray, W. Chong, T. Barclay, A. Szalay, J. vandenBerg, Terascale sneakernet: Using inexpensive disks for backup, archiving, and data exchange, arXiv preprint cs/0208011.
URL https://arxiv.org/abs/cs/0208011
[43] A. W. Services, Aws snowball (2025).
URL https://aws.amazon.com/snowball/
[44] S. Beora, Google cloud's transfer appliance: A beginner's guide (2025).
URL https://medium.com/%40santosh_beora/google-clouds-transfer-appliance-a-beginner-s-guide-e37d051c90f2
[45] J. Mannes, Amazon will truck your massive piles of data to the cloud with an 18-wheeler, in: Tech Crunch, 2016.
URL https://techcrunch.com/2016/11/30/amazon-will-truck-your-massive-piles-of-data-to-the-cloud-with-an-18-wheeler/

Matlab Code To Compare File Sizes

```
% fileSize.m
% Compare ASCII file and MAT file sizes
close all
clear all
clc

n = 1e6;
myMatf = 'myMat.mat';
myAsciif = 'myascii.csv';

u = rand(1,n)*-1;
fid = fopen(myAsciif,'w');
for ii = 1:n
    fprintf(fid,'%.17f\n',u(ii));
end
fclose(fid);
save(myMatf,'-v6','u');% uncompressed

f1 = dir(myAsciif);
f2 = dir(myMatf);
fprintf(1,'Compression = %.2f\n',f1.bytes
    /f2.bytes);
```