DATA ACQUISITION FOR THE ARROWHEAD FRAMEWORK

Mario Thron, Thomas Bangemann

Institut f. Automation und Kommunikation e.V., Werner-Heisenberg-Str. 1, 39106 Magdeburg, Germany

Abstract

The Eclipse Arrowhead Framework combines features of classical control systems and the Industrial Intenet of Things (IIoT). It was developed over several European research projects and has now reached a high degree of maturity. It is based on the basic concept of so-called self-contained local clouds and thus guarantees e.g. real-time data handling, data and system security, automation system engineering and high scalability. We present here a potential core service that enables simple and efficient data acquisition. As part of the Arrowhead Tools research project, this Extended Historian Service was developed to enable an application of data science methods to historical and real-time data in an easy and efficient way. It supports interoperability with classical fieldbuses and other data sources and can be parameterized with a graphical configuration tool.

Keywords: *IIoT, Industry 4.0, Arrowhead Framework, data acquisition, data analytics*

1. INTRODUCTION

Efficient data analysis and also artificial intelligence methods form the basis for the continuous optimization of modern industrial plants. However, the prerequisite for this is the acquisition of the necessary data. This turns out to be complex, since many communication networks and communication protocols exist in real industrial plants.

Fig. 1 shows this using the semiconductor industry as an example. The outstanding communication standard in this industry is the SEMI Equipment Communication Standard (SECS). It defines communication via serial interfaces and their message coding based on a Generic Equipment Model (SECS/GEM). The industry's ideal goal is for all equipment to communicate using this standard, making the collection of status and process data from automation systems potentially very simple.

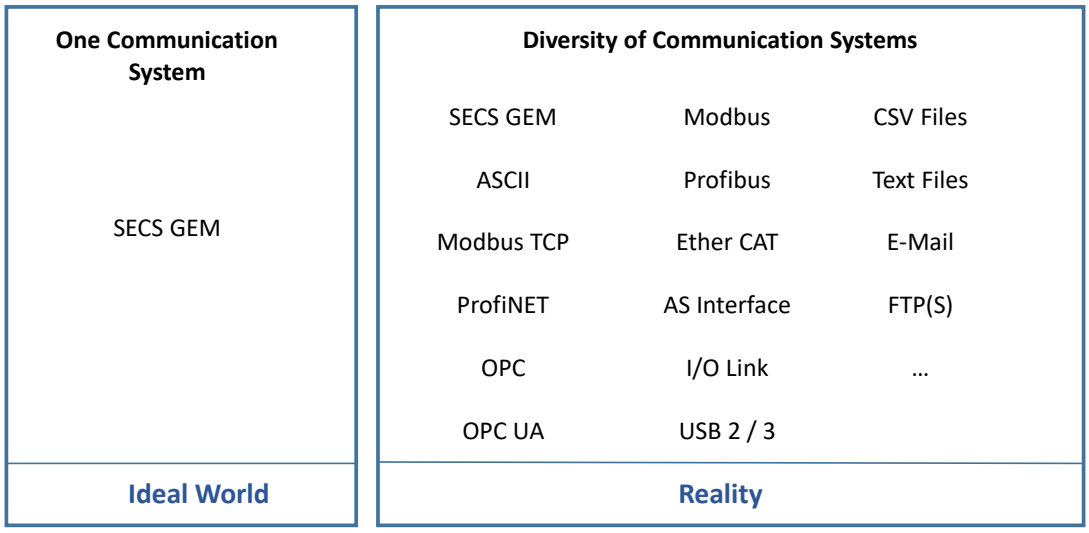
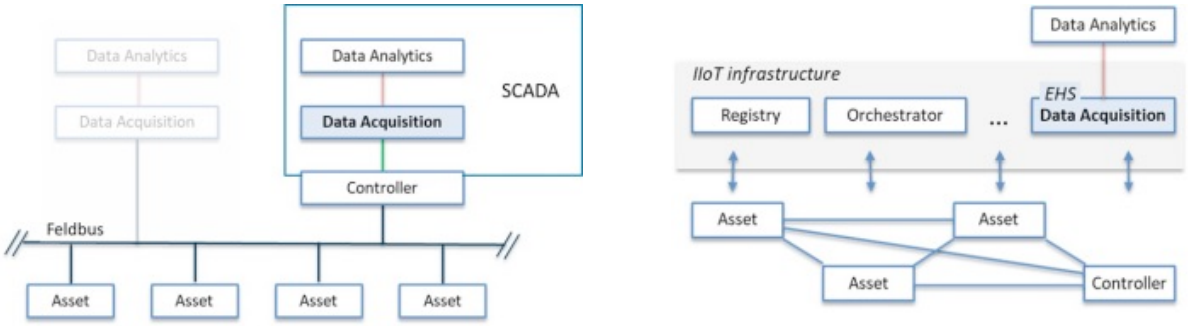


Fig. 1. *The ideal and the real world of communication systems in industrial environments, here related to the semi-conductor production.*

In reality, however, many tasks in logistics and transport technology, packaging systems, monitoring equipment and quality assurance systems are performed by technical equipment that uses other communication standards. This simply results from their cross-industry application. For this purpose, communication standards of the automation industry (e.g. Modbus TCP, ProfiNET or I/O Link) and office communication (e.g. e-mail, CSV files or data transfer via FTP(S)) are used.

Fig. 2 also shows two general trends in the collection and evaluation of data. On the one hand, there is the strictly hierarchical classical approach (Fig. 2.a). In this, a SCADA system (Supervisory Control and Data Acquisition) takes over the connection to fieldbus systems and the sensors and control systems connected to them. IIoT-based systems (Industrial Internet of Things) do not have this strictly hierarchical structure (Fig. 2.b). Instead, the services, behind which measurement, control and regulation systems can also be located, appear on a relatively equal basis. However, all services have a relationship to a set of mandatory infrastructure services. In this article, it is argued that this could and should include a service for data acquisition.



a) Classic Data Acquisition *b) Modern IIoT –based Data Analytics*

Fig. 2. *Different variants of data acquisition and data analytics systems.*

One of the modern IIoT-based communication systems to consider for data acquisition is the Eclipse Arrowhead Framework (https://projects.eclipse.org/projects/iot.arrowhead), which has been developed in several European research projects (see Delsing, J et. al. 2017). It currently has a service to get and store event data but it lacks a system for pulling data systematically and according to a schedule as it is needed for several kinds of data analytics. Thus, we propose to introduce such a system, which we call the Extended Historian Service.

Many companies do not employ full-time mathematicians or AI experts. However, their knowledge is immensely important for efficient optimization of production. In many cases, what then remains is for manufacturing companies to collaborate with universities, research institutions and other companies that offer services in the field of data analysis and artificial intelligence. However, this results in an organizational problem. The experts are not always present, but they have to adjust some parameters and learn about the data collected with the changed parameters. Remote access to the data acquisition system is necessary. But self-built solutions often lack secure communication, which is a necessary requirement for the Extended Historian Service concept.

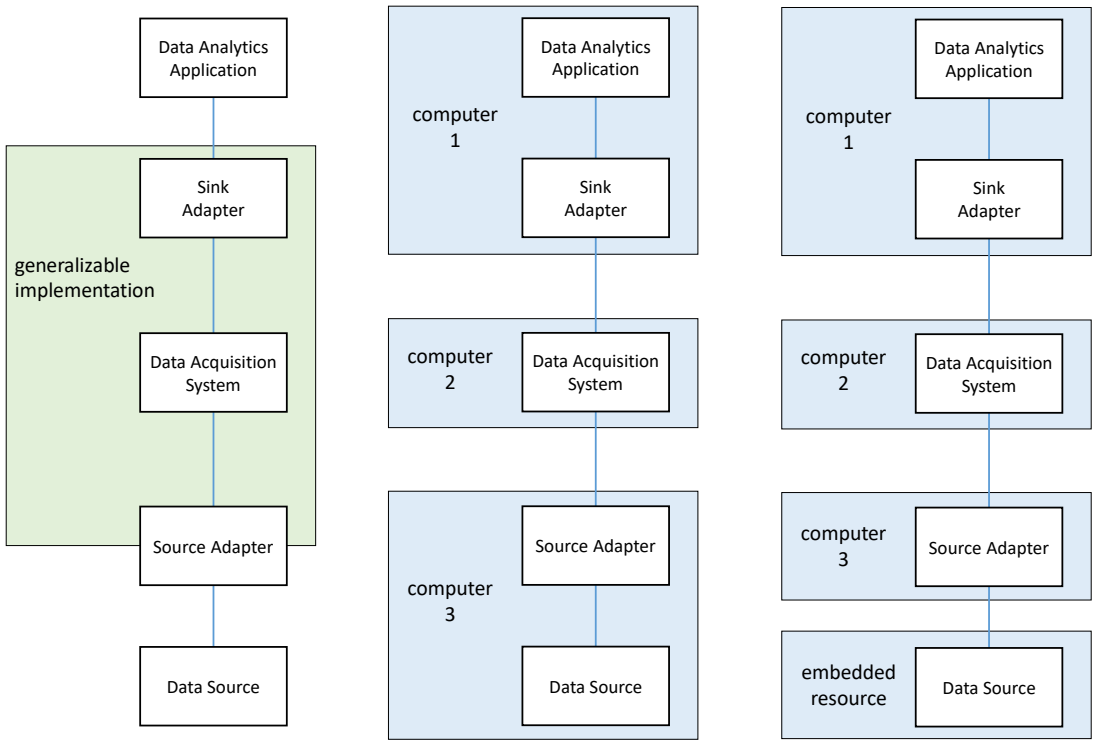
2. MATERIALS AND METHODS

2.1. General Methods

In distributed systems there are repetitive implementation parts. These can be generalized in form of software libraries or made available as service components. At which point which concept is used is of crucial importance for the user friendliness of a software system.

Fig. 3.a shows which generalizable software components exist for a data acquisition and data analysis solution. At the center is the data acquisition system. It captures the data provided by the data sources and stores it. The data analytics application accesses and evaluates the stored data. These three systems are decoupled from each other via adapters.

Data sources are connected to communication systems. Their diversity has already been discussed in the introduction. Therefore, it is not possible to implement a single source adapter that covers all these possibilities. Instead, however, a component can be developed that handles the partial function of communication with the data acquisition system. Communication with the actual data source must be supplemented on a project-specific basis. The diversity of data analytics applications is more limited. Today, many research institutes work e.g. with GNU R or with Python and various software libraries as statistical software and a lot of literature is available (e.g. Nelli, F 2018 or Jafari, R 2022). Therefore, the Sink Adapters should also be offered as elements of a software library. The Data Acquisition System is expected to be a complex system and has to work independently at runtime. Therefore, this part of the software should be implemented as an independently running service. In summary, besides writing the data analysis application, little effort needs to be invested in writing a half-source adapter, if necessary, when a new communication system is to be used.



a) Implementation strategy *b) General deployment* *c) Embedded data source*

Fig. 3. *Implementation strategy and deployment variants.*

Fig 2.b and 2.c show different variants of the deployment of the software components. In addition, there are further possibilities. In these two cases, the data acquisition system is assumed to run on a dedicated computer (24/7/365 operation). The data analysis application is certainly a software system on a personal computer. This should also provide the software library to access the data acquisition system. The two solutions differ in deployment on the source side. Fig. 2.b shows a complex data source that requires its own computer (e.g., an image analysis system). In this case, the Source Adapter can be instantiated on the same compute resource. In case 2.c, on the other hand, the Data Source is, for example, a simple sensor or control system running on an embedded resource. Here it is necessary to provide a separate

computer for the Source Adapter. This can be a small computer like a Raspberry Pi, which also can provide multiple Source Adapters for different Data Sources.

2.2. Concrete Architecture of the Extended Historian Service

In the research project Arrowhead Tools (https://tools.arrowhead.eu/home/) an Extended Historian Service (EHS) was developed. Fig. 4 provides an overview about the architecture of the EHS and its application context, which is dominated by the infrastructure components of the Arrowhead System:

- Service Registry: the service registry is used to make the EHS known in the Arrowhead local cloud.
- Authorization System: the authorization system is used to validate access to the EHS by sensors (push scenario) and to get access of the EHS to the sensors (pull scenarios) in cases, where the sensors are themselves Arrowhead application services.
- Orchestration System: the orchestration system may initiate data transfer from sensors to the EHS in cases, where the sensors are themselves Arrowhead application services.

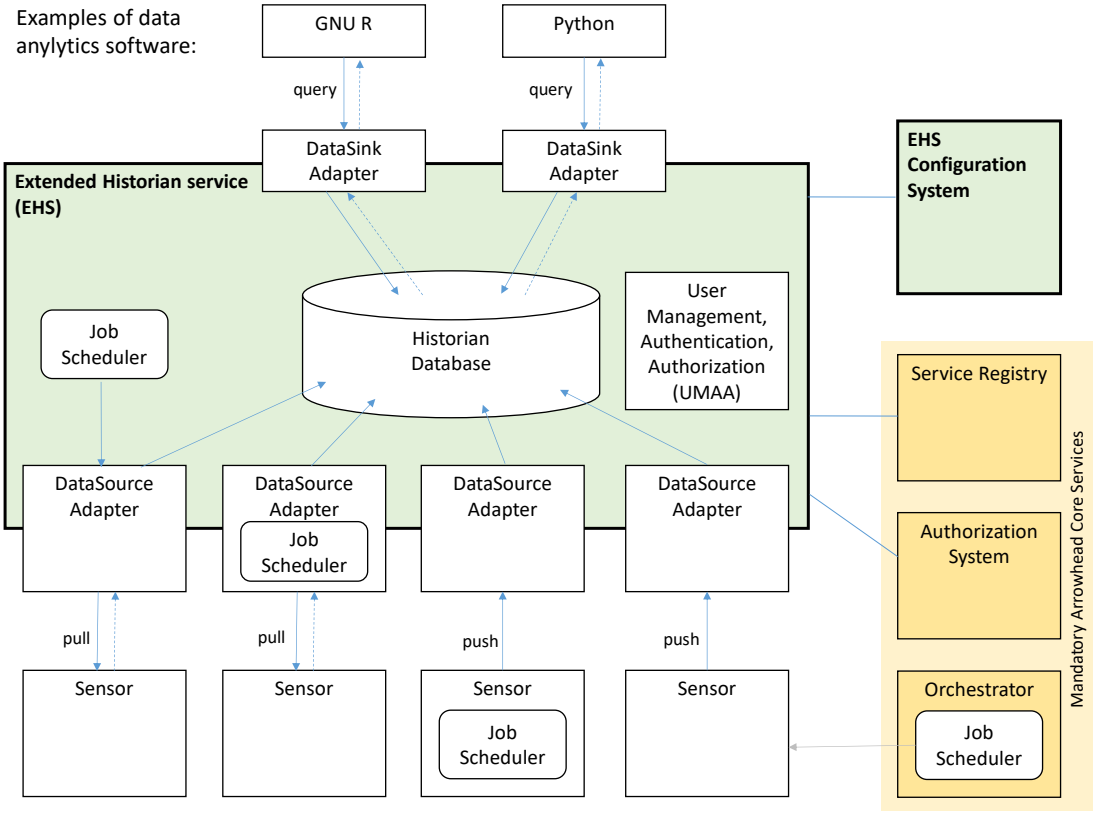


Fig. 4. *Architecture of a framework for the efficient data acquisition in a landscape of diverse industsrial communication systems – the Extended Historian Service.*

The EHS itself consists of the following components:

- Historian Database: the component, which stores the incoming data.
- Job Scheduler: the component, which is able to trigger data source adapters to gather data from sensors and to put them into the Historian Database.
- UMAA-System: a component providing functionalities for role based user management, authentication and authorization (UMAA).

The EHS is accompanied by several applications and adapters, which organize data exchange with Arrowhead and non-Arrowhead data sources or sinks:

- EHS Configuration System: this is a graphical user-frontend for configuration of the internal data model and the job-scheduler of the EHS as well as for configuration of the data source and sink adapters.
- Data Source Adapter: a component, which connects to a sensor or other data source. It is able to send and receive data over the industrial communication system used by the sensor. Fig. 4 depicts the following possibilities for initiating data acquisition:
    - The EHS job-scheduler triggers the data source adapter to get data from the sensors and to push it to the Historian Database.
    - The data source adapter triggers the data acquisition from the sensor and pushs the measured values to the Historian Database.
    - The sensor actively triggers the data source adapter to take over the data from the sensor (push) and to put it into the Historian Database.
    - The orchestrator triggers the sensor to push data to the data source adapter and to put it into the Historian Database.
- Data Sink Adapter: a component, which connects the Historian Database to the data analytics software. It is able to provide the data in a data format and over a network connection, which is usable for the analytics software.

2.3. Technology and implementation decisions

The main operation of the EHS is an infinite loop (running until a stop event occurs). It will wait for new data to be pushed and itself, it will cyclically observe the scheduling rules, whether it is necessary to pull data. But also, adapters may run in an infinite loop in order to act as a network service. Therefore, and because of the reasoning about beneficial deployment, we decided to implement the EHS and its adapters in separate processes.

This makes inter-process communication necessary. After evaluation of different technologies (Apache Thrift, gRPC, Java Threads, RMI, HTTP(S) based REST services), we decided to use gRPC (google Remote Procedure Calls) for separation of adapters and EHS. gRPC is a high performance solution, which enables to write several components in their own language (e.g. C, C++, Java, C#, Python, …) instead of being restricted to Java technologies. It supports HTTP2, data encryption via SSL/TLS and beside method invocation also stream-processing. Compared to REST services, gRPC is roughly 7 times faster (see Fernando, R 2019).

The EHS Configuration System has been implemented by using the PySide library, which makes the Qt library available in Python (see Fig. 5). Originally there was the plan for an implementation as a Web application, but the development cycles for modern Web frameworks are very short and would have consumed too much maintenance efforts after the project time.

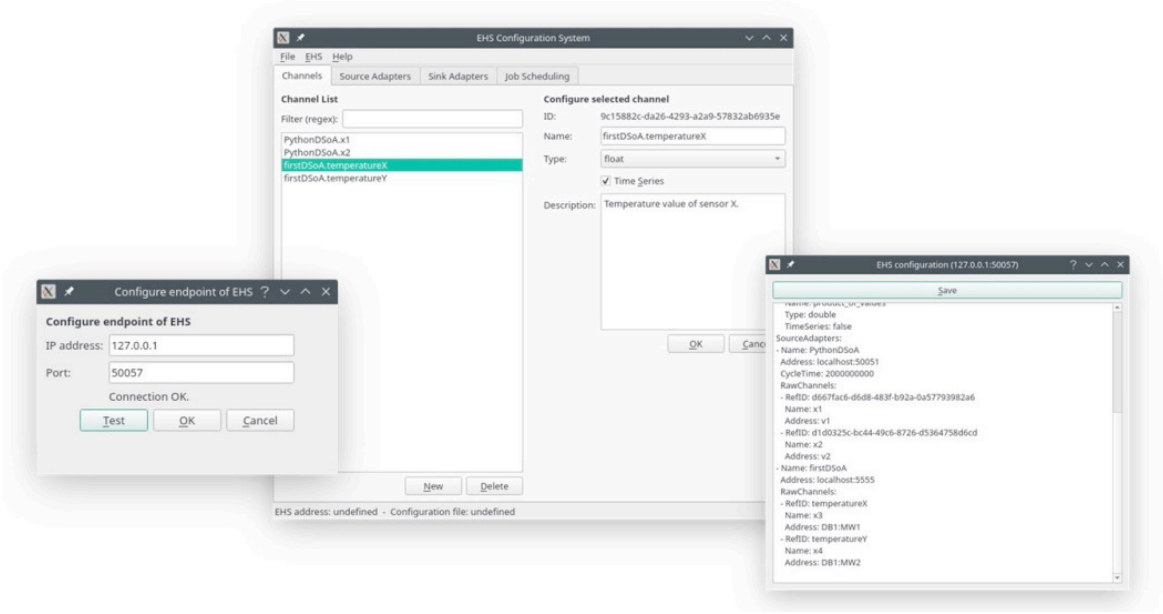


Fig. 5. *EHS Configuration System, based on PySide and Qt libraries.*

3. RESULTS

The results of our work in the Arrowhead Tools project are the concept and first implementation of a data acquisition system that supports necessary security standards and is easy to configure and extend. It will soon be released as an open source project under MIT license (https://github.com/ifak-prototypes).

The currently used security concept of the EHS is based on SSL/TLS based encryption and authentication provided by the gRPC technology. The whole communication between computer 1 and 2 and between computer 2 and 3 in Fig. 3 is encrypted. The following scenario is supported by scripts of the Open Source software for authentication:

- A companies system administrator creates a self-signed certificate a public and a private key.
- The administrator signs certificates for servers and clients (EHS and its adapters).
- Both, servers and clients can verify at run-time, wether they are authorized to access each other.

```
header:
  title: Application
  description: A data analytics application using the EHS.
client:
  address: "127.0.0.1"
  port: "50051"
  ca_cert: ../ca/cert/grpc/ca.pem
  client_cert: cert/grpc/client.pem
  client_key: cert/grpc/client-key.pem
```

a) EHS and security configuration

```
import os
import time
from ehs.ehsx import EHSClient

# initialize some variables
script_dir = os.path.dirname(os.path.abspath(__file__))

# initialize EHS
ehs = EHSClient(script_dir)

# get information about channels
channels = ehs.get_channels()

# get current values
values = ehs.get_values([channels[0], channels[1]])

# get time series
now = time.time_ns()
ten_minutes_before = now - 10*60*1_000_000_000  # ... nanoseconds based
histories = ehs.get_histories([channels[0], channels[1]], ten_minutes_before, now)
```

b) Python script to access sensor values and time series

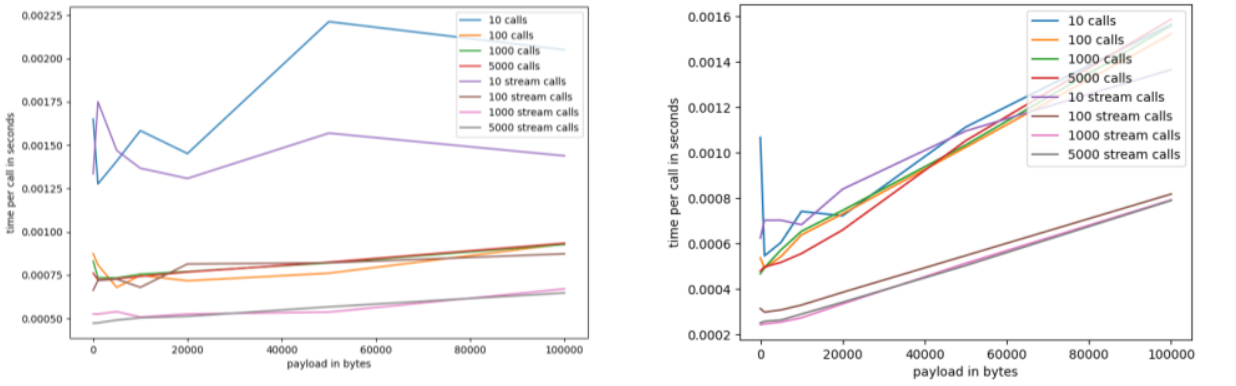
Fig. 6. *Programming interface for the data analyst to access multiple time series from the EHS*

The analysts can focus on work with values and time series with minimum effort as depicted in Fig. 6. In line 5 and 6 of Fig. 6.a the analyst needs to address the EHS and in lines 7 to 9 the certificates, which can be created automatically, are referenced. The Python script imports in line 3 a prepared EHSClient class, which can be instantiated by providing the directory, where the configuration file is situated. Then it is very easy to access current sensor values (line 15) or time series of those sensors (lines 18 to 20).

4. DISCUSSION AND CONCLUSIONS

While we reached all goals with regard to usability, we also made some research with regard to performance of our gRPC adapter communication. We compared the gRPC communication with regard to different payload sizes, amount of service calls in a rush on different operating systems.

Fig. 7.a shows results on a Linux machine, which was virtualized by use of Oracle VirtualBox. The other diagram in Fig. 7.b was result of research on a non-virtualized Microsoft Windows 10 machine. In both cases we figured out, that the more calls we do in a rush, the smaller is the time per call. On Windows we had a higher influence on the payload size than on Linux. The total time per call was lower on Windows in general, but we didn't made research with regard to the operating system virtualization.



a) Tests on localhost (Linux guest, Oracle VirtualBox VM)

b) Tests on localhost (Windows 10, native)

Fig. 7. *Programming interface for the data analyst to access multiple time series from the EHS*

Another effect could be important for installations with need for high performance: gRPC can be used in two modi. The first modus is a simple remote procedure call. The second modus is a stream call, comparable with the WebSocket technology. A single call of a client can be answered by the server with multiple responses in a row. Also other streaming models are supported. The results show, that streaming leads to higher performance.

We implemented a streaming variant of the adapters, but it showed, that our implementation became unstable due to some bugs on our side. Further work is necessary to improve the stability of the software.

ACKNOWLEDGMENTS

The research work leading to these results has received grants as project Arrowhead Tools from the European H2020 research and innovation programme and the ECSEL Joint Undertaking under grant agreement no. 82645 as well as from the German Federal Ministry of Education and Research (FKZ 16ESE0359). We are grateful to have been given that opportunity for our research activities.

REFERENCES

1. Nelli, F 2018, "Python Data Analytics: With Pandas, NumPy, and Matplotlib", *Apress Media, LLC.*
2. Jafari, R 2022, "Hands-On Data Preprocessing in Python: Learn how to effectively prepare data for successful data analytics", *Packt Publishing*.
3. Delsing, J et. al. 2017, "IoT Autmation – Arrowhead Framework". *CRC Press.*
4. Fernando, R 2019, "Evaluating Performance of REST vs. gRPC", on-line: https://medium.com/@EmperorRXF/evaluating-performance-of-rest-vs-grpc-1b8bdf0b22da, Apr 3, 2019. (last viewed 2020-02-26)