Data Handling for PLC-based Research Facilities - How to Interact With Data?

Dennis Marschall*, Nicola Bergs*, Daniel Sept*, Michael Butzek*, Nikolaos Margaritis*, and Ghaleb Natour*†
*Engineering und Technologie (ZEA-1), Central Institute of Engineering, Electronics and Analytics (ZEA)
Forschungszentrum Jülich GmbH, 52425 Jülich, Germany
†Welding and Joining Institute, Faculty of Mechanical Engineering
RWTH Aachen University, 52062 Aachen, Germany
email: [d.marschall, n.bergs, d.sept, m.butzek, n.margaritis, g.natour]@fz-juelich.de

***Abstract*—The simple access to and long-term storage of measurement data at research facilities play an important role and represents an essential basis for scientific work. There are various solutions for data acquisition, such as the Experimental Physics and Industrial Control System (EPICS) Archiver or the TANGO Framework, which are particularly suitable for use with large scientific devices. For small to medium-sized Programmable Logic Controller (PLC) controlled systems, it is often not worth the effort of such a framework, so that there is often still a lot of room for improvement. This paper describes the development of a database-based acquisition system that offers the researcher easy access to the measurement data, while ensuring good usability. In addition to data storage, the system has the ability to transfer data like parameters or test routines to the PLC for execution.**

Keywords-data acquisition, plc-controlled system, long-term storage.

I. Introduction

In research facilities, the data storage of small to medium-sized systems and test setups is often based on text files. The measurement data generated in the Programmable Logic Controller (PLC) can either be written directly to a Comma-Separated-Values (CSV) -file in the PLC [1] and stored on the file system, or it can be read out by third-party software products [2][3], such as LabView© or Matlab© and then stored as a file. The resulting text files are often transferred from the PLC to the data storage or analysing device using portable mass storage devices or File Transfer Protocol (FTP) communication. Past events, such as the best-known attack on PLC control systems with the Stuxnet worm, which was initiated via an USB stick [4], have shown that the use of portable mass storage devices on control systems and plant networks should be avoided. In various research facilities, the use of mobile data storage devices such as USB sticks is even generally prohibited. Also, the classic FTP exhibits security deficiencies owing to its age and intended use [5]. Although some of these issues have been addressed through extensions like Secured FTP (SFTP), it is advisable to check the protocol utilization within the plant network.

Large-scale facilities, such as particle accelerators or telescopes, can make good use of existing software frameworks such as Experimental Physics and Industrial Control System (EPICS) or TANGO [6]. In addition to the general functionalities of a Supervisory Control and Data Acquisition (SCADA) system, EPICS also offers the option of saving measurement data via the Archiver, exporting it or filtering it directly with own written analysis code [7]. These platforms are scalable to small systems, but the effort required to set up and maintain such a system is not negligible. In addition, such a system can only be used by researchers who have advanced knowledge of how to operate the system. Direct use by researchers who have not worked with the system before is therefore unlikely.
The following paper presents the implementation of a solution that is somewhere between the simplest file-based solution and the existing but too complex systems. In addition to safety and availability aspects, good usability by the researcher is a top priority in order to minimise the training period and avoid operating errors. The solution offers the opportunity to easily access the system's measurement data and also provides test routines in the form of parameter files which are then available in the control system.

In Sections II the basic idea and system concept is described. Section III then discusses the identified use context of the system, before the detailed technical implementation is described in Section IV.

II. System Concept

The fundamental idea of this concept is to combine the best aspects of the three components PLC, Human Machine Interface (HMI) and Databases in order to increase the scalability and adaptability for different scientific devices, while minimising the resulting issues. This is achieved by using a different approach once a component exhibits significant deficiencies in a specific area or is unsuitable for the task. While PLC and HMI Systems (in particular those produced by the same manufacturer) work well together when it's about high reliability and operation of scientific devices, they have limitations in complex user interactions and the provision of measurement data in different ways and formats. Especially when the data must be retained for an extended period and the queries to select the data are highly complex. In such cases, customised web-based systems with specialised Databases offer significant advantages. The system is designed to enable the initial contact with the researcher via a customised web application (web app). The researcher can upload the desired experiment, while being able to use convenient software such

as Excel tables to describe the different stages. This is the used by the PLC and HMI System to execute its control mechanisms. Following completion of the experiment, the researcher can interact with the web app to specify which data in which format is required. A challenge was to find the right balance between the enhanced adaptability through the use of web app and database, while maintaining the integrity of the PLC control process.

III. Ease of Use

In order to achieve good usability with the system which was to be developed, the project was started with a context analysis in which the 22 Deutsche Akkreditierungsstelle (DAkkS) guiding questions [8] were used in a revised version to carry out a semi-structured interview with the researchers. A significant challenge was the wide range of applications for scientific devices and systems. In order to find a homogeneous user group, we focused on research facility that are usually operated and managed by just a few or even one person, as the use of large SCADA systems is often not possible for these types of systems. These facility were PLC controlled plants, laboratory setups or scientific devices operated at the Research Centre Jülich.

The analysis resulted a core task, shown in Figure 1 to upload a previously defined parameter file and to download the relating measurement data again in the same way after execution. These task can be divided into 4 interactive steps, which are described in the following.

A new experiment always starts with defining the test steps of the scientific device and documenting them, for example, in the form of an Excel table or a CSV file. This file is referred to below as the parameter file and contains general data and system parameters such as operating points, holding times, priorities and dependencies. Fields can be defined which are only filled in by the PLC during execution. These can be timers, counters, comment fields, or other variables available in the PLC.

The parameter file can then be uploaded in the second step inside the Data Management Portal (DMP) (Figure 3) using the upload parameter file function. During the upload, a plausibility check of the parameter file is carried out, which can be configured for the specific facility. This configuration can contain area exceedings of working points, missing working points, duplicate step numbers or incorrect priorities. In case of an error inside the file, the upload is denied by the DMP and the user receives an error list to revise their parameter file. If the parameter file does not contain any errors, it is saved to the database. This process generates an unique experiment ID, which is shared with the DMP user during the upload. If the researcher wants to make changes inside the uploaded parameter file before starting the file in the research facility, the file can be modified by upload it again via the DMP. The portal uses the general data from the file to check whether it is a new file or an existing one. If the file already exists, the database entry will be updated.

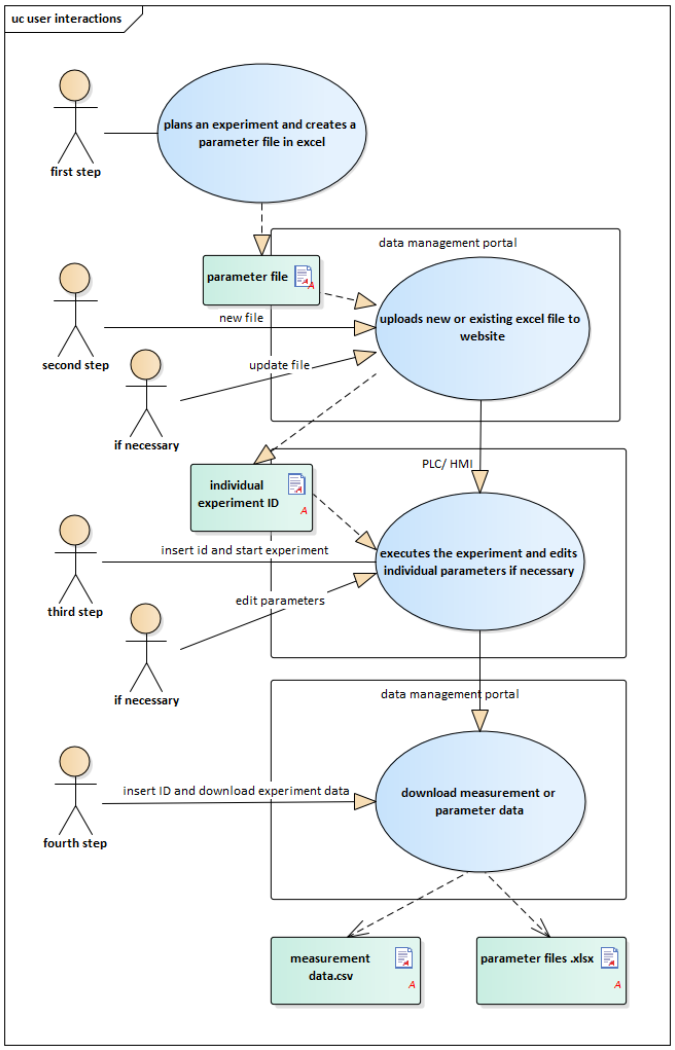

Fig. 1. User interactions.

The third step of the process describes the actual execution of the experiment. In contrast to the previous preparatory steps, this step must be carried out on the HMI of the facility. By entering the experiment ID in the HMI system, the PLC accesses the database and loads the parameter file into the temporary memory. The individual steps can be viewed in the HMI system and edited until they are executed by the PLC. During the experiment, the PLC creates a copy of the parameter file as a log file. In this log file, existing fields to be completed by the PLC are then filled in automatically. This can be, for example, the start time, end time or the hold time of an individual step. A comment function is included within the HMI system. If a comment is written, this is also recorded in the log file with a timestamp, so that the researcher can easily and safely document events during operation. After the test has been completed, the log file is automatically saved in the database. In addition to the log file, all measurement data are also saved cyclically in the database, whereby the cycle time can be selected and adapted to the measuring application.

If an active experiment is running, the cyclic data is tagged with the experiment ID to ensure better identification. If no active test is running, the cyclic data is logged with the tag ”no active test” and a timestamp. Step four describes that the researcher wants to access the measurement data and log files, after the experiment has been completed. This is again done via the DMP, access to the research facility or the HMI system is no longer required. To access the dataset for the experiment, one can enter the experiment ID. Use this ID to directly access the data or apply filters to retrieve specific information, as illustrated in Figure 3. The selected data can be downloaded as a CSV file using the download button. A distinction can be made between the German and English CSV format in order to fit to the different evaluation software products. This means that the experiment can be analysed by the researcher independently of the research facility. The data remains stored in the database after downloading so that the data can be accessed again at any time via the DMP. In addition to the cyclical measurement data, the parameter log file can be downloaded the same way.

IV. IMPLEMENTATION

Figure 2 shows the general structure of the system. The main component is a database for all data and parameters as a Single Source Of Truth (SSOT). This is connected to the PLC, the HMI system and the developed data management portal (DMP) webserver, which is described in the following subsections.

A. *Database*

Due to the desired scalability and adaptability of the system to different scientific devices, the Not only Structured Query Language (NoSQL) database MongoDB is used, which is not restricted to a fixed data schema like Structured Query Language (SQL) Databases [9]. The used structure which resembles the often used JavaScript Object Notation (JSON) -format and the possibility to optimise collections for use with time series data strengthens this decision. Alternative databases which also have time series functionality would be for example InfluxDB.

The current hosting solution is realised using a local MongoDB [10] docker container on a QNAP Network Attached Storage (NAS) model TS-473A with 4*4TB Disks in a Raid10 Setup. This leads to a higher output-rate and could tolerate up to two disk failures in the best-case.

Due to the docker setup, it is possible to host this solution on every machine which is able to run a docker environment. Search speed for queries is improved using the time series functionality and indexes for certain fields. The NAS is currently doing daily snapshots of the necessary folders, which can be manually saved on an external drive for later restoration. The saved data can also be kept on the NAS-System, this is only limited due to the available amount of disk space and the lifespan of the disks.

An experiment simulating four machines producing the same dataset (about 150 values with integers, boolean values and floating point numbers) every second for about 12 years used about 26 GB of disk space for the entire MongoDB data folder.

B. *PLC based database connection*

*1) **Connection**:* To establish the functionalities for system communication, it is essential to utilise function blocks from Beckhoff Automation GmbH & Co. KG (Beckhoff) libraries within the PLC. Building upon these components, the engineering process involves crafting appropriate advancements, to expand the functionality of the building blocks to create a system with working communication between the PLC and the database. To enable access through the PLC, various settings are required. The chosen database must be initialised using the TF6420-Database Server [11] extension tool from Beckhoff and a connection must be established via the IP address in the required network. The created database contains an ID for target identification in later workflow stages. This number cannot be dynamically assigned by the user, instead, it is incremented numerically. A ”Standard Connection String Format” is selected as the communication basis, meaning that data and queries from the PLC are sent and received in string format, structured into a JSON format within this string. In the realm of communication, the TwinCat 3 (TC3) Database Library [12] is employed for tasks such as inquiries or filter configurations. For the formatting of data into JSON strings or the conversion of these strings into structures, the TC3 JSON XML [13] is essential. For example, to retrieve data already present in the database, the following process is required. Using the Query-Builder Function block with the included functionality of QueryOptionDocumentDB [12], a search in the database is triggered via the PLC. A string must be generated containing the desired search information, to serve as filter for exploring the database. This serves as a filter and is passed to the Query Find function. To ensure a correct access to the desired data, the search location in the form of the Collection Name as a string must be passed to the block. To initiate a request, the ExecuteDataReturn [12] method of the QueryBuilder module can be employed to submit a query to the database. The ID of the desired database needs to be selected in addition to the filters and collection settings. Upon the successful execution of a request, signifying the retrieval of the intended data, the data can be formatted based on user preferences using methods from the fbNoSQLResult [12] module. This can be done either by converting it into a string (fbNoSQLResult.ReadAsString) [12] or, alternatively, directly into a predefined structure within the PLC (fbNoSQLResult.ReadAsStruct) [12]. The string is more flexible but also more complex, as desired individual data must be extracted and assigned to a PLC variable. With structure formatting, this happens automatically. In this context, it is essential to ensure an exact match in the structural configuration between the PLC and the database entry. The data must be absolutely identical in their structural format; otherwise, the transfer would going

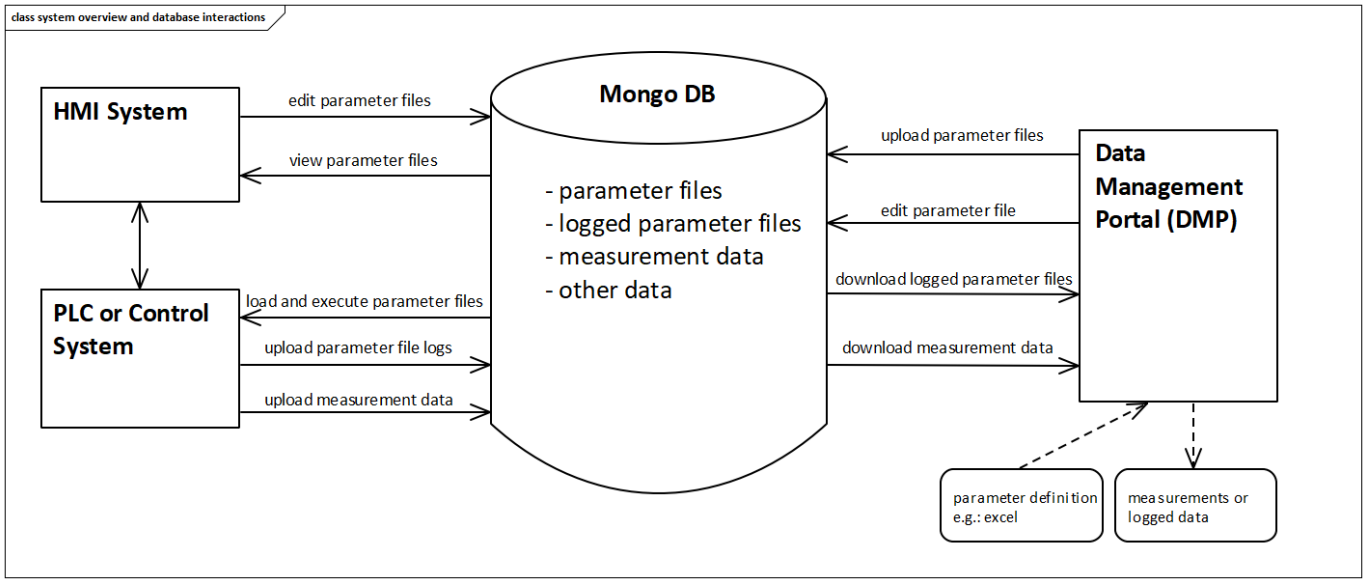

Fig. 2. System overview and database interactions.

to fail. The process for writing data and information utilised the same functionalities as the request process. Only the query function is switched to QueryOptionDocumentDB Update [12] or QueryOptionDocumentDB InsertOne [12], depending on the application's requirements.

*2) **Problems and Limitations**:* To establish a functional communication system, different challenges must be addressed. The definition of filters in a JSON format occurs at the user level and must precisely match to a pattern. Deviations lead to communication errors, making troubleshooting more difficult, especially for complex filters. Information written to the database must be in JSON format. Beckhoff provides a block "fbJsonDataType.AddJsonValueFromSymbol" [13] to automate this task and store the result into a string. The standard size of strings is 255 characters. However, this is often not sufficient, so this range must be declared manually larger. This leads to problems with greater data records. Changes, such as ISO date adjustments, in this string may render it unable to be graphically represented. Errors in conversion or subsequent edits may not be visually inspected, complicating the identification process. The mentioned example process occurs whenever timestamps are a factor. The PLC operates with its own date time type, which does not fit with the ISO date type. In cases where specific specifications are necessitated within the string, such as executing a unit conversion for the database, manual string manipulation becomes complex. Beckhoff offers a range of fundamental functions in its TC2 Utilities library [14], including find, insert, or delete functions. These functions serve to locate and modify essential positions, variables, or information elements. Nonetheless, it is worth noting that the procedure becomes notably time-consuming when dealing with larger volumes of data (length of the string). The temporary storage of information for preceding or future processing can occupy a significant amount of internal memory due to the string's size. This may negatively impact PLC task and cycle times. The communication between the PLC and the database does not operate at the cyclic speed of the PLC, leading to delays in processing data and issues with cyclic logging intervals. Due to the strict arrangements regarding variable declarations and structure setups, the flexibility provided by the database, with its NoSQL system, is limited via the PLC. It therefore does not exploit the full potential of the selected database type.

C. Data Management Portal

The DMP, shown in Figure 3 is realised using a website with two structurally separated "Apps" to interact with the user, shown in Figure 3. One App is responsible for uploading and downloading parameter files, while the other is used for interacting with measurement data, including providing various CSV formats. A more detailed explanation of the different options can be found in Section III. The website is build using the Django Framework [15] which is python based and most designs are realised using the Django Template Language (DTL) and the Bootstrap Framework [16].

This subsection provides a more detailed explanation of the DMP. It outlines the processes involved in validation, the Database used and how the communication with the DMP works. Additionally, it identifies potential issues and limitations.

*1) **Validation**:* The validation is used every time a parameter file is uploaded. It is realised with the Frictionless Framework [17], which provides tools to validate, transform and enrich data. Structures called schemas are written inside a JSON Document to specify the requirements for the parameter files. It can specify constraints like minimum, maximum, field

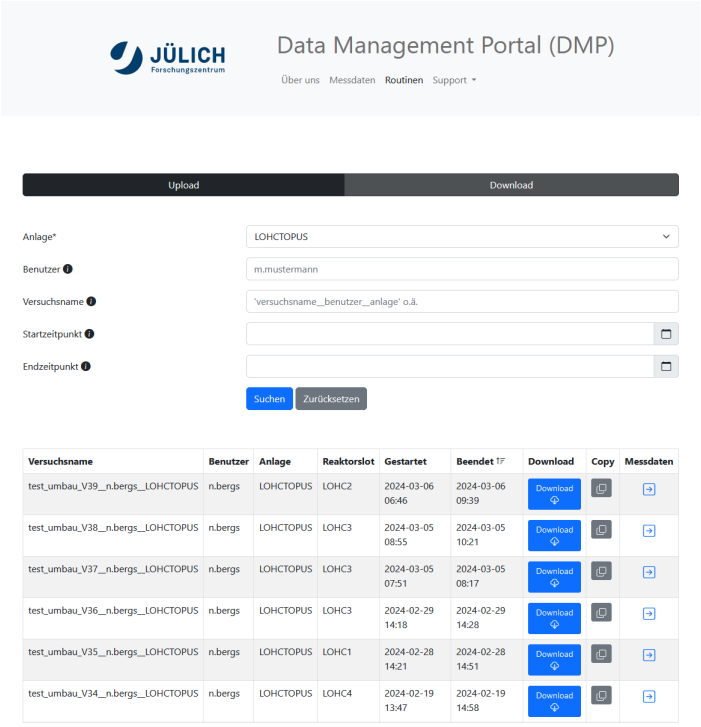

Fig. 3. DMP - upload and download app.

types and flags for requirement of fields. Later on, transformation pipelines can also be specified. Once the parameter file does not conform to the schema, the user will receive a comprehensive list of errors and violated constraints to rectify them.

*2) **Database**:* Django by itself doesn't support MongoDB. Its language (python), which is often used in the scientific community, and the amount of additional usable packages compensates for this fact. This leads to a faster adaptability of the software. In order to be able to use the Django Queries for simple models, we use the Djongo Mapper [18]. More complex and nested structures like the time series datasets are queried using the PyMongo Library [19].

*3) **Queries**:* Each page in the DMP has a form similar to Figure 3 where the user can restrict or expand the search results by specifying time, creator, experiment ID and other parameters. These search parameters are passed to the Django backend, where a query is dynamically generated. This query is then sent to MongoDB and the results are processed and returned to the user. In the default case, with no search parameters, the results are capped to reduce query time.

*4) **Problems and Limitations**:* By using the DTL and the main principles of Django, most of the website content is created in the backend and sent completely to the user. This approach can lead to longer waiting times if there is a high workload, like long computations or transformations. The concept also reaches its limits when a lot of interactivity is needed. This would need further use of JavaScript or a JavaScript Framework like Vue.js [20]. Although the DMP ensures that the parameter files are validated and correct, it is possible to corrupt the data via the PLC or HMI connections if used incorrectly.

D. HMI based database connection

In addition to the PLC, the HMI system can also access the database directly. This is done via a server extension written in C# on the HMI server. The extension uses the MongoDB Driver and the MongoDB Binary JSON (BSON) data type. A query of the database for all available object IDs with the corresponding experiment names can thus fill a list element or a combo box with the parameter files available in the database. The complete entry of the experiment ID can be simplified by selecting the corresponding list element from the query. The database connection can be used to load complete parameter files into the temporary memory of the HMI server so that they can then be edited using the graphical tools available in the HMI. To save the changes, the temporary data set can then be uploaded back into the database. HMI-side access to the database always makes sense, if the operation or visualization can take place independently of the PLC. In this way, the limitations of PLC-based database communication described in Section IV-B2 can be minimised.

V. CONCLUSION AND OUTLOOK

The developed concept for data handling in research facilities covers the interaction principles of the ISO 9241-110 [21] in many cases due to the user-centred approach. Suitability for the user's tasks is the focus of the entire concept in order to avoid unnecessary steps and ensure easy access to the measurement data.

MongoDB as central data storage offers great flexibility and controllability due to the large number of interfaces. In addition to the integration of the DMP, the PLC and the HMI included in the developed concept, various other software tools and hardware platforms can of course be connected to the system. The hardware requirements to connect the MongoDB database are very low, so that a communication-enabled ESP32 board can already be connected to the database to serve as a data logger.

The realised validation of uploaded parameter files, described in Section IV-C1, protects new system users from operating error and helps them to comply with system limits. Forms and task-related help files, stored in the DMP, guide new users through the use of the tool. An iterative development process with different prototypes, which was carried out through expert-based usability evaluations with researchers, has led to a user interface that meets user expectations.
Further tests are pending following the launch of the system. It is planned to carry out observational user tests with "Thinking Aloud" by new, inexperienced users. As the concept can be implemented in various systems with only minor adjustments, the total number of DMP users will add up in a long-term perspective so that usability can also be measured in future using tools such as the SUMI [22]. This was previously not possible due to the individual systems with only a few users and offers further potential for improvement.

The analysed user needs also result in further extensions for the future such as the DMP can only be used to store parameters, as well as measurement data and make them easily accessible. A graphical representation of the data, which has so far only been implemented inside the HMI system via its own data memory, would be a potential extension of the DMP. The integration of plot or dashboard environments or even the direct evaluation of measurement data using a Jupyter notebook [23], which is integrated in the DMP, can further increase the usability of a scientific device and lead to consistent evaluation results, especially with changing researchers. However, any extension must take into account the scalability of the system and its adaptability to different scientific facilities. Depending on future requirements, the individual components of the systems could be extended or improved. For example, a 3-2-1 Backup Strategy [24], use of the Django authentication system or adaptations to achieve high availability and redundancy could be implemented.

REFERENCES

[1] Beckhoff Automation GmbH and Co. KG, "TE1000 TwinCAT 3 PLC-Library: Tc3 System", December 2023.

[2] Beckhoff Automation GmbH and Co. KG, "TE1410 TwinCAT 3 — Interface for MATLAB®/Simulink®", September 2022.

[3] Beckhoff Automation GmbH and Co. KG, "TF3710 TwinCAT 3 — Interface for LabVIEW™", February 2024.

[4] M. Baezner and P. Robin, "CSS Cyber Defense Hotspot Analysis Stuxnet", Version 1 ETH Zürich, October 2017.

[5] D. Springall, Z. Durumeric and J. A. Halderman, "FTP: The Forgotten Cloud" 2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 2016.

[6] C. Böhme, C. Deliege, M. Simon, M. Thelen, V. Kamerdzhiev, R. Modic and Ž. Oven, "Challenges of the COSY Synchrotron Control System Upgrade to EPICS" 2024 19th Int. Conf. Accel. Large Exp. Phys. Control, Cape Town, South Africa, 2023.

[7] L. R. Dalesio, M. R. Kraimer and A. J. Kozubal, "EPICS Architecture", Argonne National Laboratory Argonne, New Mexico, January 1991.

[8] DAkks Deutsche Akkreditierungsstelle, "Leitfaden Usability Version 1.3 [Usability guide version 1.3]", 2010.

[9] R. Deari, X. Zenuni, J. Ajdari, F. Ismaili and B. Raufi, "Analysis And Comparision of Document-Based Databases with Relational Databases: MongoDB vs MySQL," 2018 International Conference on Information Technologies (InfoTech), Varna, Bulgaria, 2018.

[10] 'MongoDB', https://www.mongodb.com/ [Online], Accessed: 2024.03.06.

[11] Beckhoff Automation GmbH and Co. KG, "TF6420 TwinCAT 3 Database Server", December 2023.

[12] Beckhoff Automation GmbH and Co. KG, "TF6420 TwinCAT 3 PLC-Library: Tc3 Database", December 2023.

[13] Beckhoff Automation GmbH and Co. KG, "TE1000 TwinCAT 3 PLC-Library: Tc3 JsonXml", May 2022.

[14] Beckhoff Automation GmbH and Co. KG, "TE1000 TwinCAT 3 PLC-Library: Tc2 Utilities", November 2023.

[15] "Django Project", https://www.djangoproject.com/ [Online], Accessed: 2024.03.05.

[16] "Bootstrap Library", https://getbootstrap.com/ [Online], Accessed: 2024.03.05.

[17] "Frictionless Data", https://frictionlessdata.io/introduction/ [Online], Accessed: 2024.03.05.

[18] "Djongomapper", https://www.djongomapper.com/ [Online], Accessed: 2024.03.05.

[19] "PyMongo 4.6.2 documentation", https://pymongo.readthedocs.io/en/stable/ [Online], Accessed: 2024.03.05.

[20] "Vue.js", https://vuejs.org/ [Online], Accessed: 2024.03.05.

[21] DIN-Normenausschuss Ergonomie (NAErg) , "Ergonomics of human-system interaction - Part 110: Interaction principles (ISO 9241-110:2020); German version EN ISO 9241-110:2020".

[22] J. Kirakowski and M. Corbett, "SUMI: the Software Usability Measurement Inventory", British Journal of Educational Technology, 1993.

[23] "Project Jupyter Documentation", https://docs.jupyter.org/en/latest/ [Online], Accessed: 2024.03.06.

[24] "What is a 3-2-1 Backup Strategy?", https://www.seagate.com/de/de/blog/what-is-a-3-2-1-backup-strategy/ [Online], Accessed: 2024.03.05.