ISSN: 2832-790X

DOI: 10.33552/OJRAT.2024.02.000546

Online Journal of
Robotics & Automation Technology

Iris Publishers

Review Article

Blueprint and Proof of Concept of an IOT Cloud Platform for Brownfield Production Facilities

Ing Juergen Goehringer and M Eng Josef Fleischmann*

Research Associate at the Institute of Digital Production Management (IDPM), Germany

***Corresponding author:** M Eng Josef Fleischmann, Research Associate at the Institute of Digital Production Management (IDPM), Germany

Received Date: December 15, 2023

Published Date: April 26, 2024

Abstract

This document presents a comprehensive study on the design and implementation of an IoT Cloud System architecture tailored for brownfield industrial applications. Brownfield environments, characterized by their existing legacy systems, pose distinct challenges for integration with advanced IoT technologies. The primary objective of this research was to conceptualize a flexible and scalable system architecture that not only accommodates but also enhances the capabilities of such environments through IoT solutions.

Through the application of the proposed architecture in various industry projects, the research demonstrates the system's ability to adapt to diverse operational requirements, including real-time data processing, error analysis, and seamless interaction with existing machinery. The successful deployment in sectors such as machine tool automation and beverage handling systems validate the architecture's robustness and adaptability.

Introduction

In the domain of industrial engineering, brownfield production facilities constitute a formidable frontier for digital integration. These sites, often characterized with legacy systems, present a unique set of challenges to the adoption of Industrial Internet of Things (IIoT) paradigms. The implementation of IoT cloud platforms within such established infrastructures is critical, not just for technological conformity but for the substantial operational enhancements and competitive advantages they promise. The potential of IoT-enabled platforms to catalyze real-time data exchange, predictive analytics, and autonomous decision-making processes underscores the urgency of this digital migration [1]. Digitization of brownfield facilities is essential to elevate operational efficiency through improved asset utilization and downtime reduction, achieved via continuous monitoring and predictive maintenance protocols. This is facilitated by IoT cloud platforms, which offer scalable and flexible solutions necessary for incremental modernization. The transition enables seamless data integration from heterogeneous sources, empowering legacy systems to interface with advanced data processing tools, thereby invigorating the decision-making with actionable insights [2].

However, retrofitting these production environments with cutting-edge IIoT capabilities comes with challenges. The task of digital retrofitting involves tailoring IoT cloud solutions that respect the constraints and capitalize on the intrinsic value of existing infrastructure. The design and deployment of such IoT cloud platforms necessitate meticulous research and strategic planning to navigate the complexities of system compatibility, cybersecurity, and data integrity. Accordingly, the ensuing article delves into the systematic implementation of an IoT cloud platform specifically tailored for brownfield applications. It scrutinizes the dualistic challenge of integrating state of the art communication protocols such as OPC-UA or MQTT, as well as legacy protocols such as Modbus TCP or even RS232 [3].

Approach

Our methodology is predicated on devising a robust IoT cloud platform architecture suited for the complexities of brownfield applications. Initial efforts concentrate on formulating the system's backbone, delineating the server and gateway structures essential for seamless integration and data management. The approach advances through the practical realization of this architecture within industrial environments, evidenced by field implementations. Conclusively, the paper synthesizes the empirical data, appraising the platform's efficacy and providing critical reflections on its deployment.

Design of the System Architecture

The design of the IoT Cloud system architecture ensures comprehensive adaptability, accommodating cloud, on-premises, and local installations to meet the multifaceted demands of industrial operations. Cloud deployments offer scalable computing power and global connectivity, ideal for expansive data-driven environments [4]. On-premises installations address enterprises' needs for security and data sovereignty, often required in tightly regulated industries. Local setups are tailored for sites where real-time processing is paramount and internet access may be unreliable or unavailable, ensuring that critical operations continue without disruption.

The architecture must facilitate the operation of the IoT Cloud in all presented modes of operation. Regardless of the mode chosen, the IoT Cloud consists of five critical software components that interact through defined protocols. These components and their relationships are visualized in Figure 1.

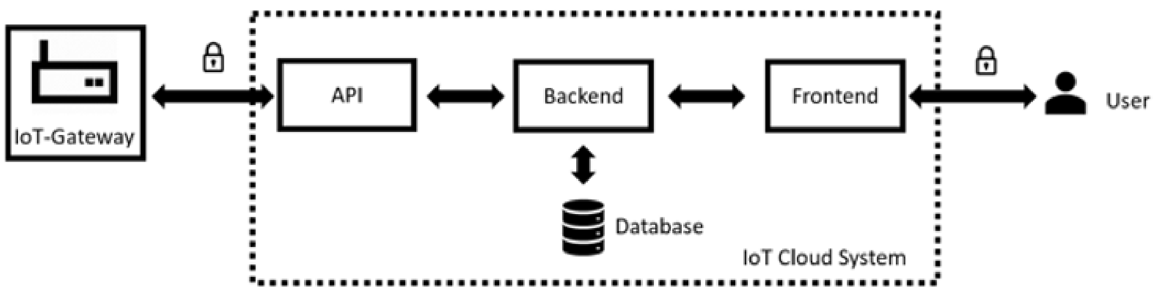


Figure 1: IoT Cloud Architecture.

The schematic indicates that the IoT Cloud constructs an integrated system with the backend, frontend, and database at its core. Interfacing with the external world is orchestrated via an API that connects to the IoT Gateway and a frontend that interfaces with users. Given the sensitive nature of data traversing these channels, encryption is imperative to safeguard communications. The backend is the central logic unit within the IoT Cloud, coordinating the system's overall functionality. In the following sections, we will delve into the detailed explanation of the communication protocols that secure data transfer within the system and the external interfaces. Furthermore, the server configuration will be outlined, showcasing its critical role in cloud operations, data handling, and system management. Lastly, the gateway architecture will be detailed, illustrating its function as a conduit between the cloud platform and the physical devices within the industrial setting.

Communication Model

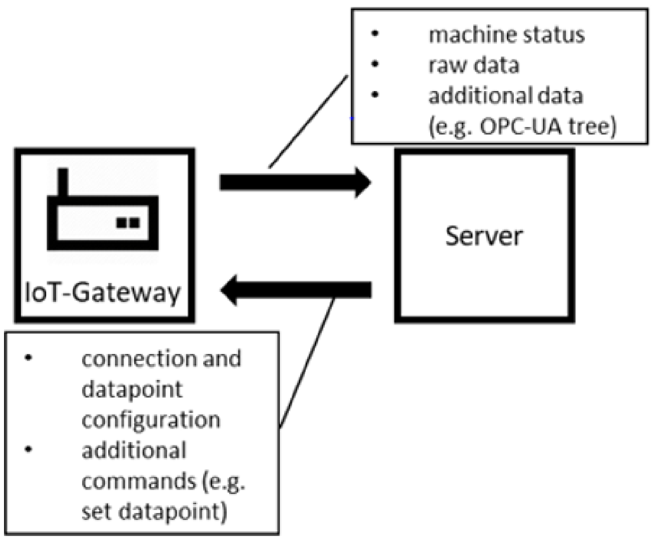

machine status
raw data
additional data (e.g. OPC-UA tree)
IoT-Gateway
Server
connection and datapoint configuration
additional commands (e.g. set datapoint)

Figure 2: Information exchange between IoT Gateway and Server.

Within the IoT Cloud system, communication for certain components is predefined: database communication occurs via the respective database protocol, while communication between the backend and web-frontend conventionally occurs over Hypertext Transfer Protocol (HTTP). The API, an integral part of the backend, manages the exchange of information between the IoT-Gateway and the server. The communication between the client (web browser) and the frontend is secured through HTTPS, which is a standard method to ensure a secure data exchange with web applications. However, the nature of the communication between the IoT Gateway and the API remains to be determined. To select an appropriate communication medium and protocol for this channel, it is necessary first to understand the type of information being exchanged. This information is depicted in Figure 2.

The IoT Gateway needs to convey various connection parameters (address, port, authentication, etc.) and the variables to be read to the server so that it can communicate appropriately with the machinery. Based on these parameters, the gateway captures the data from the machines and sends it back to the IoT Cloud.

The choice of communication model can be either Request/Response or Publish/Subscribe. In the Request/Response model, a client initiates a request which is followed by a server response, and after each exchange, the connection is terminated. On the other hand, the Publish/Subscribe model, via a broker, allows for continuous sending and receiving of messages between publishers and subscribers, as depicted in Figure 3 [5].

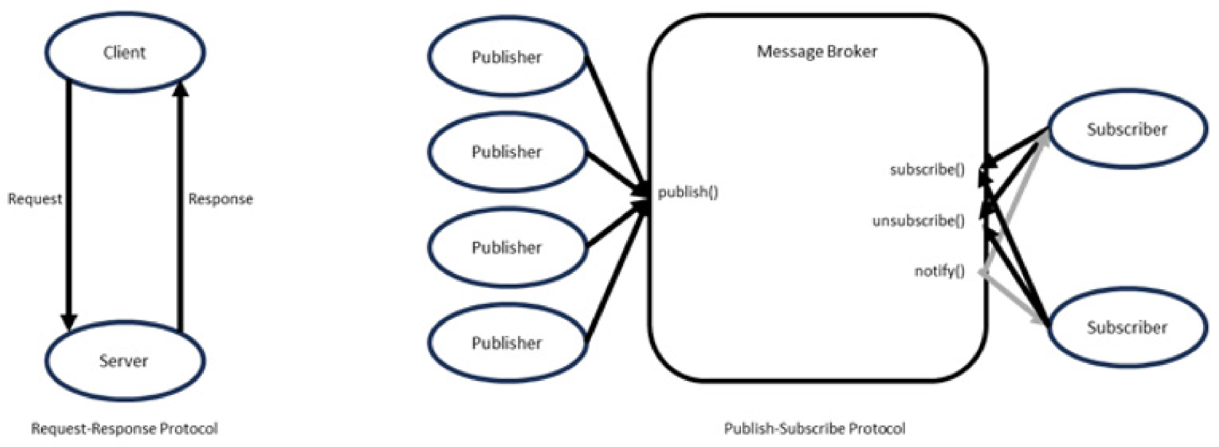


Figure 3: Difference between Request-Response and Publish-Subscribe.

Given the dynamic nature of the data and the need for a flexible, scalable communication method, the Publish/Subscribe model appears to be the more suitable choice. It allows for the IoT Cloud to act as a subscriber for machine data and as a publisher for connection parameters, while the IoT Gateway reverses these roles as needed. It also allows the Server to initiate communication with the IoT Gateway, which enables executing commands on the IoT Gateway triggered by the user, such as setting datapoints on the machine. This type of communication is typically implemented using the MQTT protocol [6]. It is a publish-subscribe-based messaging protocol that offers an efficient solution for IoT communication with minimal overhead, addressing the constraints of device bandwidth and network latency. When augmented with Transport Layer Security (TLS), as in MQTTS, the protocol ensures the confidentiality and integrity of messages between devices and the cloud infrastructure.

The lightweight nature of MQTTS, alongside its low power and bandwidth usage, makes it well-suited for the IoT Cloud system, particularly in industrial contexts where network efficiency is critical. The protocol's support for persistent sessions and message retention policies further enhances its applicability for reliable message delivery in machine-to-machine communication. Thus, the IoT Cloud system will use HTTPS at the user interface and MQTTS for communication with the IoT Gateway, ensuring an end-to-end secure data transmission.

Server Architecture

For the data flow within the application to be set up in a performance-oriented and scalable manner, the architecture of the backend requires further elaboration. Assuming a state where the customer has fully configured the system (connection parameters, data points, etc.), the following processing procedure underlies the incoming data: Figure 4

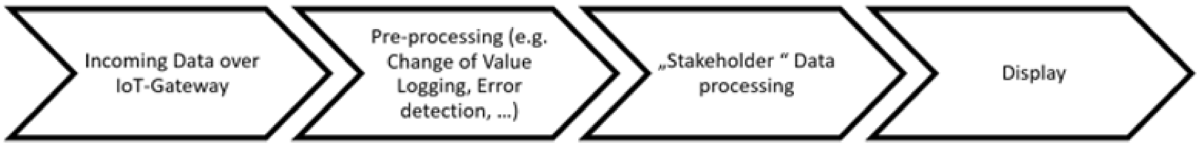

Incoming Data over IoT-Gateway
Pre-processing (e.g. Change of Value Logging, Error detection, ...)
„Stakeholder " Data processing
Display

Figure 4: Data flow in the IoT Cloud System.

The general term “stakeholder” refers to any functionality within the IoT Cloud System for which updated data at the control level are relevant. The “stakeholders” may change both at runtime (when certain functionalities are configured) and over the course of further software development. However, it is certain that there must be a base functionality that informs the so-called “stakeholders” when data have been updated. In conjunction with the previous requirements, a possible structure of the backend is shown in Figure 5:

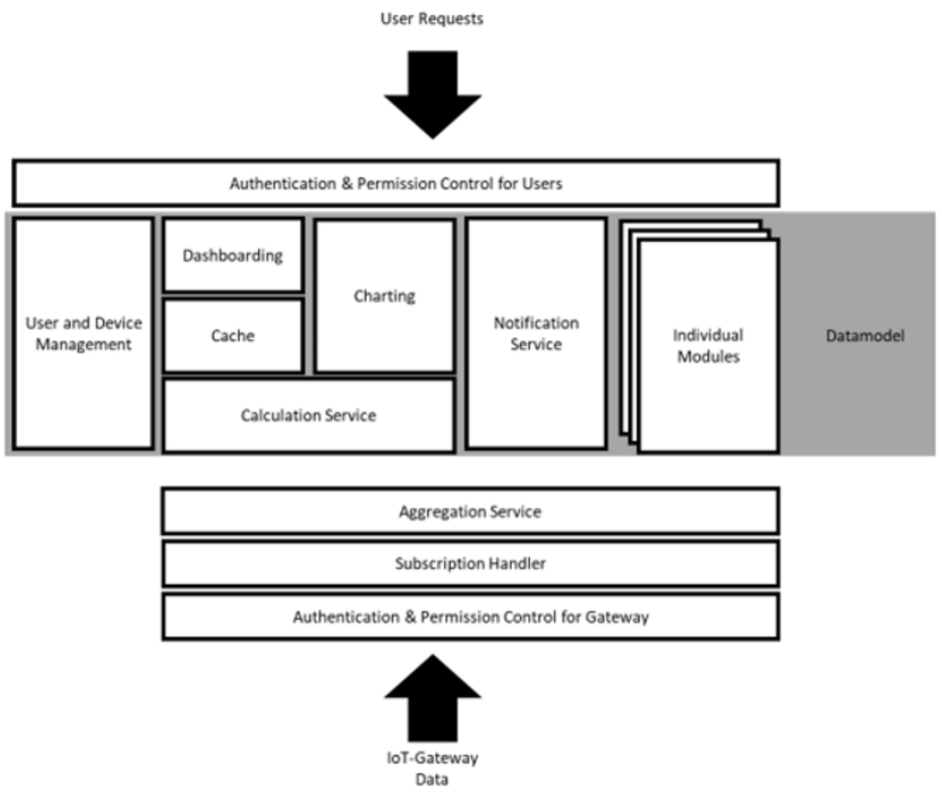


Figure 5: Backend Architecture of the IoT Cloud System.

Before processing requests from users and IoT Gateways, it is verified whether the request is legitimate. This involves determining whether the client is authenticated and whether it is authorized to access the requested or provided data, that is, whether the data falls within its organizational domain. All modules for which updated data from the gateway are relevant must register with a “Subscription Handler.” This registration specifies which data are to be subscribed to and which function is to be executed with the updated data. With this system, arbitrary modules can subscribe to data and implement an individual processing logic based on it. The Subscription Handler underpins the computation and notification service, as well as individual modules for specific customers. Within the computation service, data are further aggregated (e.g., averaging, division by operational states, etc.) and passed on to dashboarding and charting modules. Charting refers to user-specific evaluations that a user has configured at a particular time. These need to be calculated individually in “real-time.” Dashboards, however, are configured evaluations that are the same for all users within the organization. To reduce the loading time and computing effort of the dashboards, a cache precedes the dashboarding.

The management modules (e.g., users, controllers, data points, etc.) are not connected to the Subscription Handler, as they do not require real-time values. All modules have access to the data model, allowing them to link further data from the database as required.

Gateway Architecture

The IoT Gateway is designed to concurrently connect with and manage multiple devices. To enable this functionality, it is necessary to initiate a separate thread for each device. Above these individual device threads, an additional master thread is required to communicate with the IoT Cloud System, possessing the capability to independently start and stop the device threads [7]. This consideration leads to the architecture illustrated in Figure 6:

The GatewayHandler is a thread that communicates at regular intervals with the IoT Cloud System via the API and provides it with current device data. It receives information from the IoT Cloud System about the devices and the variables that need to be read. For each device, the GatewayHandler initiates a DevConnector and passes on the details concerning the connection parameters and variables to be read. The DevConnector is a generic class that must be specifically implemented for each communication protocol. The functions to be implemented are illustrated in Figure 7:

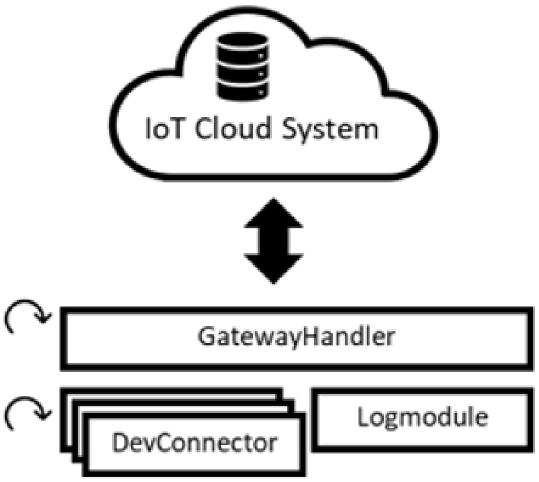
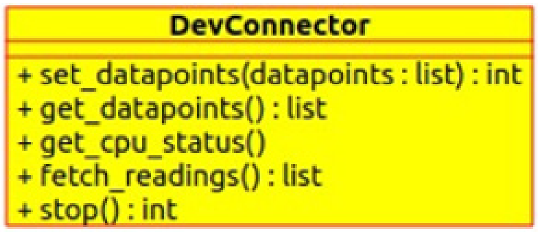


Figure 6: IoT Gateway Architecture.



```
DevConnector
+ set_datapoints(datapoints : list) : int
+ get_datapoints() : list
+ get_cpu_status()
+ fetch_readings() : list
+ stop() : int
```

Figure 7: DevConnector Methods.

The GatewayHandler is a thread that communicates at regular intervals with the IoT Cloud System via the API and provides it with current device data. It receives information from the IoT Cloud System about the devices and the variables that need to be read. For each device, the GatewayHandler initiates a DevConnector and passes on the details concerning the connection parameters and variables to be read. The DevConnector is a generic class that must be specifically implemented for each coThe methods listed in Figure 7 must be specifically implemented for each communication protocol to ensure that the interface for the GatewayHandler remains consistent regardless of the device. The GatewayHandler periodically invokes the functions fetch_readings () and get_cpu_status (), aggregating and transmitting this data back to the IoT Cloud System. Additionally, the IoT Gateway incorporates a LogModule, which records all errors, warnings, and informational messages related to communication. This logging facility is crucial for troubleshooting and determining the root cause in the event of errors.

Application

Leveraging the core functionality of the IoT Cloud system, contact was established with various companies to deploy the software in a real-world context and to further its development based on practical applications. The following chapters describe the resulting projects.

Manufacturer of Loading and Unloading Systems for Machine Tools

The project partner is a manufacturer of automation solutions for loading and unloading machine tools.

Project Description

The project goal was the integration of the software into the partner's own production system to collect data from their production facilities and to link it with production data from the factory data collection (FDC) and enterprise resource planning (ERP) systems. The aim was to make the production operation transparent, to determine the utilization capacity of the machines, and to assess the progress and efficiency of specific orders.

The challenge in this project lay primarily in integrating the IoT Cloud system into a heterogeneous environment. Two manufacturing islands with two plants each were identified for integration, all providing data via different communication protocols. In addition, the software had to be enabled to import data from external systems.

Enhancements to the IoT Cloud System

The project was divided into three phases, which could be largely worked on independently:

- Integration of machines

- Connection to third-party systems (FDC and ERP)
- Creation of a custom dashboard

The phases are explained in further detail below:

Integration of machines

The four machines communicate via OPC-UA, SIEMENS S7 proprietary, ProfiNET, and FOCAS2 protocols. The IoT Gateway already had the necessary drivers for both OPC-UA and the S7 proprietary protocol at the start of the project, as they are common and easy to test against.

ProfiNET:

For the ProfiNET protocol, there are no freely available implementations of a ProfiNET master, regardless of the programming language. However, as it is a standard in the automation industry, it is available for common PLCs. However, it would be impractical to have to use an additional device in addition to the IoT Gateway. The CoDeSys PLC runtime, however, is also available as a soft PLC for the embedded Linux system. Since the hardware underlying the IoT Gateway will run on an embedded Linux PC, it can run the soft PLC in addition to the Gateway software.

Since the data is thus read by the IoT Gateway only indirectly, via the soft PLC, the interface between the IoT Gateway and the soft PLC must be implemented. An OPC-UA server could be used in the soft PLC so that no additional drivers would be needed on the IoT Gateway side. However, since an intermediate format will also be necessary for the FOCAS protocol (described below), it can also be used here.

A text file-based format (specifically, CSV) can be easily implemented and debugged across various platforms. Therefore, it is suitable for this interface issue. The architecture for communication with formats that cannot be directly mapped in the IoT Gateway is thus structured as shown in Figure 8:

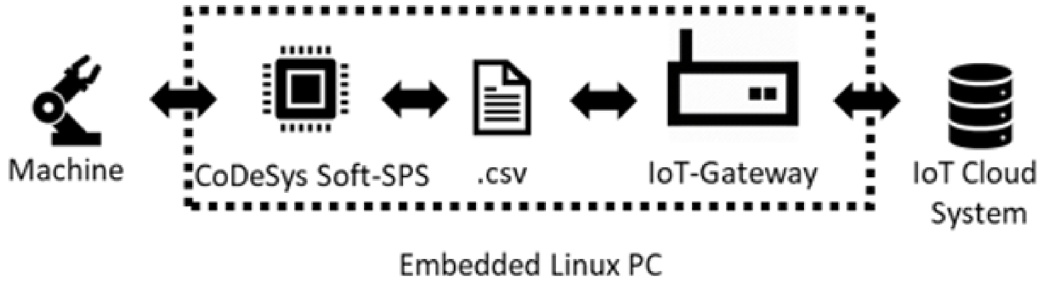


Figure 8: Architecture for connecting to non-IoT Gateway native drivers.

FOCAS:

The CNC controls manufacturer FANUC has its proprietary protocol for communication with its controls. The manufacturer provides a library for capturing data from the control unit. The library is available for the .NET environment and C/C++.

Contrary to the documentation, however, the library could not be used under Linux but required a 32-bit Windows operating system. Thus, it was necessary to implement the program for reading FANUC controls under Windows and to convert the read data into a CSV intermediate format. To avoid the need for an additional PC to run the 32-bit operating system, it was executed in a virtual machine hosted on the IoT Gateway's system. This approach serves the FOCAS protocol as follows: Figure 9

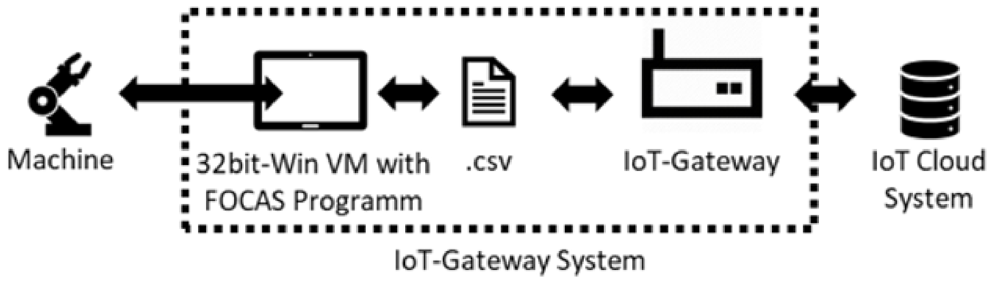


Figure 9: Architecture for connecting to FOCAS interface.

Connection to Third-Party Systems

Both the FDC and ERP systems are underpinned by an ORACLE database. Users were provisioned for the IoT Cloud System to access production data. The IoT Cloud System is tasked with displaying the active production orders for each manufacturing island and providing supplementary background information. These orders are captured by a clocking system within the production hall and communicated to the FDC system.

Interaction and the triggering of subsequent actions are specific to each system and must be implemented as individual modules (refer to Figure 5). The IoT Cloud System discerns active or modified orders by querying the database, necessitating temporal behavior within the module to track database changes, which is facilitated by a dedicated thread. This module's architecture is presented in Figure 10.

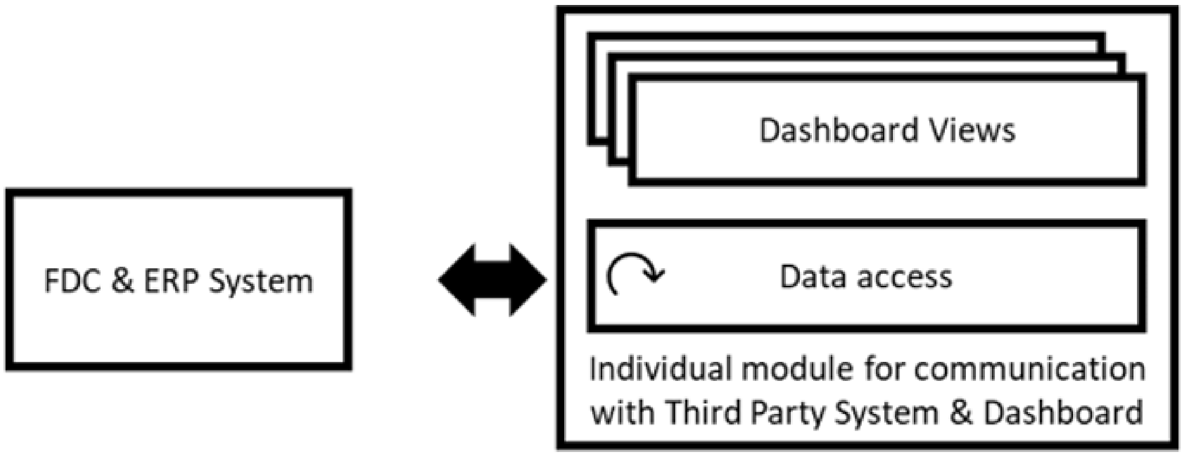

FDC & ERP System
Dashboard Views
Data access
Individual module for communication with Third Party System & Dashboard

Figure 10: Architecture individual module.

Creation of Custom Dashboards

For the development of custom dashboards based on information from the aforementioned external systems, it was first necessary to logically group individual machines into manufacturing islands. The cloud system had existing functionality to form a machine state from individual variables. This was expanded by a component named "Joined PLCs," which allows for the creation of a collective of PLCs for which a composite machine state can be calculated. Using the data from the FDC and ERP systems, along with the composite and individual machine states, dashboards were implemented that display machine times and order data.

Manufacturer of Handling Systems for the Beverage Industry

The project partner is a manufacturer of systems that sort, relocate, and transport beverage crates and pallets.

Project Description

Initial discussions with the project partner indicated that the IoT Cloud system was an appropriate solution for simplifying the commissioning process and potentially aiding subsequent error analysis in the event of malfunctions. The partner had faced challenges in tracking the sequences of logic based on sensor data and understanding how these sequences contributed to errors.

The project's objective was thus to upgrade the IoT Cloud system and the IoT Gateway to capture and analyze data within milliseconds. Moreover, it was intended for the IoT Cloud system to function in an offline mode, operating locally at the plant to ensure data remained inaccessible outside the customer's network. This requirement gave rise to the "IoT Cloud Light," optimized for performance on embedded hardware.

Enhancements to the IoT Cloud System

The IoT Gateway's drivers conventionally work in a polling mode, querying predefined variables at set intervals (e.g., every 500ms). However, this method can miss transient events such as voltage spikes or short-lived logic states, leading to potential data loss. PLCs internally manage variables within cycle times ranging from approximately 5 to 100ms- a rate not achievable via a polling communication link. To preserve all relevant data, a proprietary protocol that operates on a push principle was implemented in cooperation with the project partner. This protocol allows the PLC to send data as it becomes relevant, facilitating processing within the PLC's cycle time. The data is then sent through a Transmission Control Protocol (TCP) channel established by the IoT Gateway. If the data cannot be processed quickly enough through the communication channel (either by the PLC or the IoT Gateway), it is temporarily stored in a cache, ensuring no loss of information.

With the PLC initiating data transmission autonomously, the IoT Cloud system required adjustments to autonomously generate data points in the database. As a result, technicians are able to configure which data should be monitored directly within the PLC, without additional configurations in the IoT Cloud system. Further enhancements were made to the IoT Cloud system's graphing functionality to effectively analyze data within millisecond increments. This improvement incorporated a temporal navigation tool to swiftly traverse through time series data. A "Delta-Tool" was also introduced, enabling interactive identification of differences between data points.

Service Provider for Industrial Plant Maintenance

The project partner is a small business offering maintenance services for industrial plants. Their focus is primarily on retrofitting current hardware into existing facilities, a process known as “retrofitting.” They mainly cater to the print finishing industry and also handle wastewater treatment plants.

Project Description

The project involves not only the basic integration of existing plants into an IoT platform but also centers on the analysis of Overall Equipment Effectiveness (OEE) metrics for print finishing plants. The goal is to implement a configurable OEE metric within the IoT Cloud system.

Modifications to the IoT Cloud System

As the collaboration with the project partner was established towards the end of this study, the modifications to the IoT Cloud system have been largely conceptual. The implementation of the OEE analysis is based on planned functionality for metrics. Given that a universal application of these metrics is resource-intensive, the intention is to develop a specific but configurable OEE evaluation for print finishing plants, demonstrating the capability to potential stakeholders.

Conclusion

The conceptualization and deployment of the IoT Cloud System architecture within brownfield environments affirm its technical viability and architectural flexibility. The diverse industry applications explored underscore the architecture’s adaptability to distinct brownfield conditions, validating the design’s efficacy in heterogeneous operational contexts. The empirical evidence gathered from the deployment across varying industrial settings indicates that the system architecture can accommodate the nuanced demands of legacy systems while facilitating modern IoT functionalities. It successfully integrates real-time data processing capabilities and interfaces with pre-existing hardware, reaffirming the importance of adaptability in system design for industrial applications.

This study contributes to the body of knowledge by demonstrating a scalable and modular approach in adapting IoT Cloud Systems to brownfield applications, providing a framework for future research and development in the field of industrial system digitalization. The architecture’s capacity to assimilate with specialized use cases paves the way for further exploration into custom module development and integration strategies tailored to the unique landscapes of legacy industrial operations.

Acknowledgement

None.

Conflict of interest

No conflict of interest.

References

1. TA Tran, T Ruppert, G Eigner, J Abonyi (2022) Retrofitting-Based Development of Brownfield Industry 4.0 and Industry 5.0 Solutions, IEEE Access 10: 64348-64374.
2. MH u Rehman, I Yaqoob, K Salah, M Imran, P P Jayaraman, et al., (2019) The role of big data analytics in industrial Internet of Things, Future Generation Computer Systems 99: 247-259.
3. WPL ,Chi-Hung Hsiao (2021) OPIIoT: Design and Implementation of an Open Communication Protocol Platform for Industrial Internet of Things, Internet of Things Vol 16.
4. MAO Pessoa, MA Pisching, L Yao, F Junqueira, P E. Miyagi et al., (2018) Industry 4.0, How to Integrate Legacy Devices: A Cloud IoT Approach. in IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society, Washington, DC, USA.
5. B Wukkadada, K Wankhede, R Nambiar, A Nair (2018) Comparison with HTTP and MQTT In Internet of Things, in International Conference on Inventive Research in Computing Applications, Coimbatore, India.
6. C Sun, K Guo, Z Xu, J Ma, D Hu (2019) Design and Development of Modbus/MQTT Gateway for Industrial IoT Cloud Applications Using Raspberry Pi, in 2019 Chinese Automation Congress, Hangzhou.
7. VG Găitan , I Zagan (2021) Experimental Implementation and Performance Evaluation of an IoT Access Gateway for the Modbus Extension, Sensors, Nr 1: 246.