A Study on Optimizing Signal Path in Model Predictive Control and Fault Detection System of Three-Tank Pilot System Using Reference Architecture

Jukka Kortela[1][a], Yared Tadesse[2] and Kim Miikki[2]

[1]*Aalto University School of Chemical Engineering, P.O. Box 16100, FI-00076 Aalto, Espoo, Finland*

[2]*Bellmer Finland Oy, Vanha Messiläntie 6, 15860 Hollola, Finland*

fi *fi*

Keywords: Cyber Physical System, Industrial Internet of Things, Rami, Iira, Model Predictive Control, Three Tank System, Model, Experiment, Fault Detection, Fault Diagnosis, Parity Equations.

Abstract: This paper presents the model predictive control and fault detection and diagnosis system of a three-tank pilot within a novel cloud-integrated industrial automation framework. The system architecture includes a state-of-the-art NodeJS-based gateway facilitating communication between the cloud service and the automation system. OPC DA has not been updated to function with the latest programming libraries and operating systems, which significantly reduces the performance of automation systems. The optimized signal path through the OPC DA is developed and compared to the OPC UA tunneller implementation through experiments on a real three-tank pilot system with an industrial ABB 800xA automation system. The results demonstrate that the optimized signal path significantly reduces the control interval by a factor of 5, leading to a quicker controller response. In fault detection and diagnosis, the delay is only 22 milliseconds with an optimized signal path compared to 408 milliseconds when using OPC UA tunneler software.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

1 INTRODUCTION

Since the invention of the Programmable Logic Controller (PLC) in 1968, industrial automation has passed through milestones marked by advances in information technology. The first milestone is the linking of the PLC with the personal computer (PC) in 1986. The next main milestone was reached in 1992 with the introduction of Ethernet and Transmission Control Protocol and Internet Protocol (TCP/IP) connectivity for PLCs. The current trend in industries is the transformation from industrial Ethernet and wireless communications to advanced information technology (IT) solutions where traditional automation is merged with cyber-physical systems (CPS) combining communications, information and communication technology (ICT), data and physical elements and the ability to connect devices to one another. This transformation results in what is now known as Industrial Internet of Things (IIoT) or the 4th Industrial Revolution (Industry 4.0) where every step of a manufacturing process is interconnected. Cloud computing and data analytics are among the technologies driving the IIoT. (IEC, 2015)

According to IEEE, the term architecture in the context of information technology is ”the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution”. (IEEE, 2022)

A reference architecture in the context of information technology documents such things as hardware, software, processes, specifications and configurations, logical modules and interrelationships. According to IBM Rational Unified Process, a reference architecture “is, in essence, a predefined architectural pattern, or set of patterns, possible partially or completely instantiated, designed, and proven for use in particular business and technical contexts, together with supporting artifacts to enable their use. Often, these artifacts are harvested from previous projects”. (Evensen, 2013)

A reference architecture for IIoT serves the purpose of providing common and consistent definitions for the IIoT, its subsystems and design patterns, and a

[a] https://orcid.org/0000-0001-7831-1413

Kortela, J., Tadesse, Y. and Miikki, K.
A Study on Optimizing Signal Path in Model Predictive Control and Fault Detection System of Three-Tank Pilot System Using Reference Architecture.
DOI: 10.5220/0012995400003822
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 21st International Conference on Informatics in Control, Automation and Robotics (ICINCO 2024) - Volume 1*, pages 351-358
ISBN: 978-989-758-717-7; ISSN: 2184-2809
Proceedings Copyright © 2024 by SCITEPRESS – Science and Technology Publications, Lda.

common lexicon and taxonomy for discussing specification of implementations of IIoT.

Currently, there are several reference architectures that can be employed for deploying Factory of Future (FoF). Two of the most popular reference architectures, RAMI 4.0 and IIRA, and their interoperability will be discussed in the following paragraphs. RAMI 4.0 is the product of Industrie 4.0, which is a national project of the German government initiated in 2011 through German's Ministry of Education and Research (BMBF) and the Ministry for Economic Affairs and Energy (BMWI) (European Commission, 2017), which has since been joined by industries and academia to form a consortium called Plattform Industrie 4.0. Industrie 4.0 aims to increase value in manufacturing and decrease waste by transforming the way products are developed, produced, managed, and consumed. The project focuses on the industrial manufacturing sector and connects value chains by integrating things and processes to form cyber physical systems (CPS). The novelty of Industrie 4.0 results from the combination of already existing and new technologies such as embedded computers, intelligent sensors, mobile broadband internet access, and Radio Frequency Identification (RFID) in the industrial environment into a uniform, integrated solution through standardized communication. (VDI Verein Deutscher Ingenieure e.V., 2015) The Industrie 4.0 concept is implemented through RAMI 4.0; a service-oriented architecture that has been designed for efficient sharing of data and information between all the shareholders taking part in the Industrie 4.0 ecosystem. RAMI 4.0 (registered DIN SPEC 91345 in Germany) ensures that all participants in Industrie 4.0 share a common perspective and build a common understanding. (DIN, 2016)

The Industrial Internet Consortium (IIC) first published IIRA in the form of a Technical Report in 2015. Founded by AT&T, Cisco, General Electric, IBM, and Intel, the mission of IIC is to reach industrial interoperability and consensus on IIoT platforms. The IIC is a part of the Object Management Group (OMG) and today has 19 working groups and over 250 members of industrial and academic background. In July 2019, the latest version of IIRA, IIRA v1.9, was published by the Industrial Internet Consortium Architecture Task Group, which is a subset of the IIC Technology Working group. (Industrial Internet Consortium, 2019) IIRA is a reference architecture of what IIC calls Industrial Internet Systems (IIS). These systems are defined as end-to-end application systems for industrial tasks. They include technical components as well as interactions with users. According to the IIC, IIRA is a "business-value-driven and concern-resolution-oriented" reference architecture for the IIoT. (Industrial Internet Consortium, 2019) IIRA itself is based on the Industrial Internet Architecture Framework (IIAF), which provides basic conventions, principles and definitions. The IIAF builds on the international standard ISO / IEC / IEEE 42010: 2011 and performs basic architectural description constructs, such as Concern, Stakeholders, and Viewpoint. The viewpoints are one of the key building blocks of IIS. There are four viewpoints: Business, Usage, Functional, and Implementation. (Industrial Internet Consortium, 2019)

1.1 Industrial Data Protocols

The goal of the Industry 4.0 research initiative is to enable networked, flexible and therefore adaptive production. The challenge here is that production data is often very distributed and heterogeneous. As a solution to this, the OPC Unified Architecture (OPC UA) standard provides a context-based data description model alongside its communication protocol specification. In OPC UA, metadata is defined with standardized data models that enable a uniform understanding of the data in the value chain.

There are several technologies that provide the functionality needed to bridge the operational technology (OT) and information technology (IT) gap (Bonomi et al., 2012), such as, time-sensitive networking (TSN), cloud computing and OPC Unified Architecture (OPC UA). Although TSN has received a lot of attention, a suitable technology such as OPC UA PubSub was only recently explored for its potential to meet tight timing guarantees [5], [6]. However, an open issue is that the timing guarantees are only valid if the OPC UA PubSub is isolated from the OPC UA client-server communication model. The reason is that both communication paradigms use a shared data model, which raises issues related to simultaneous data access that need to be addressed. A client-server instance may prevent access to PubSub data or change the value during a read operation with unexpected consequences. Analysis in (Denzler et al., 2022) includes the overall RT-TSN-OPC UA concept, an analysis of common concurrent data access mechanisms for their suitability, and identifying critical code segments in the open62541 OPC UA stack.

The work in (Großmann et al., 2014) describes an approach to modeling the information of an aggregated OPC UA server to combine the representation of a single entity in different data models. The work in (Müller et al., 2022) presents a methodology for the integration of standardized information models into existing OPC UA servers. However, the problem is the overhead brought by the automatic OPC UA adapter in time-critical systems.

In (Reiswich and Fay, 2012), the authors outline a method for migrating an OPC DA (Data Access) server to the OPC UA standard. The integration of the data model existing in the DA server is done by manually mapping each of the corresponding nodes. The work in (Reiswich and Fay, 2012) is suitable for model integration in this context of paper, but its disadvantage is the lack of automatic mapping. To meet above challenges, this paper implements the OPC DA and OPC UA NodeJS libraries, which enable automatic mapping and an optimized connection to time-critical systems.

1.2 The Control of the Three-Tank System

A comparison of Model Predictive Control (MPC) with Proportional-Integral-Derivative (PID) control has been conducted in (Kortela, 2022). The performance can also be compared with Internal Model Control (IMC) using a similar interval of 100 ms, as presented in (Ujjwal Manikya Nath and Mudi, 2023). Newer methods beyond classical PID include Adaptive Safe Experimentation Dynamics (ADED) (Ghazali, 2019) and methods utilizing Nonlinear Constraint Optimization (Govind et al., 2023). Both linear and nonlinear schemes were able to achieve the primary objective of controlling the liquid level of the system. However, when choosing the appropriate scheme to apply, stakeholders must consider a trade-off between accurate system representation and the ease or cost of implementation (Samuel Emebu, 2023). In (Alhassan Osman and Arıcı, 2023), the Adaptive Pole Placement Controller (APPC) method is extended by incorporating it with a Sliding Mode Controller (SMC) to enhance system robustness. Nevertheless, in real implementations, the data transfer rate has a significant effect on performance.

The rest of this paper is structured as follows:

The optimized signal path and the three-tank system are utilized in both Model Predictive Control (MPC) and fault detection, and have been presented in Section II and Section III, respectively. Section IV presents the model predictive control of the three-tank system, while Section V introduces the fault detection method applicable to this system. In Section VI, the architecture of the proposed method is detailed. Section VII presents the test results of the MPC and fault detection system using the optimized signal path. Finally, Section VI presents the conclusions of the proposed method.

2 OPTIMIZED SIGNAL PATH WITH OPC DA

This document proposes an approach that makes it possible to make the data of an already existing OPC DA server available for an optimized OPC UA server. The structure consists of four components: Wireless Arduino MKR WiFi 1010 + Raspberry Pi 3 Model B+ (2.1), optimized OPC DA client 2.2), NodeJS DA Client (2.3), and Gateway PC + Spring framework (2.4).

Fig. 1 shows the Arduino MKR WiFi 1010, where 3 level measurements are connected to the analog inputs of the Arduino.

2.1 Wireless Arduino MKR WiFi 1010 + Raspberry Pi 3 Model B+

1. In Arduino code, the value of the tank level 1, for example, is read from the first pin with the call
   ```
   sensorValue = analogRead(A0);
   ```
2. The value is scaled by executing the command
   ```
   outputValue = map(sensorValue, 0, 1023, 0,
   255);
   ```
3. Finally, the scale and value are written to the serial port with commands:
   ```
   Serial.print("sensor = ");
   Serial.print(sensorValue);
   Serial.print("\t output = ");
   Serial.print(outputValue);
   ```
4. The OPC UA server is created by calling
   ```
   new opcua.OPCUAServer()
   ```
5. Correspondingly, to read the serial port input, an object is created with the call
   ```
   new SerialPort( { path: '/dev/ttyACM0',
   baudrate: 9600 } )
   ```
6. To read the serial port, an object is created with the
   ```
   new ReadlineParser()
   ```
   command, and writing data to the port triggers the ”on” event for the object.
7. The object ”ThreeTank” is added to addressSpace with command addObject and its component as nodeId ”Level 1” is added with command
   ```
   namespace.addVariable
   ```
8. When the ”on” event occurs, a new value is written to the ”Level 1” component.

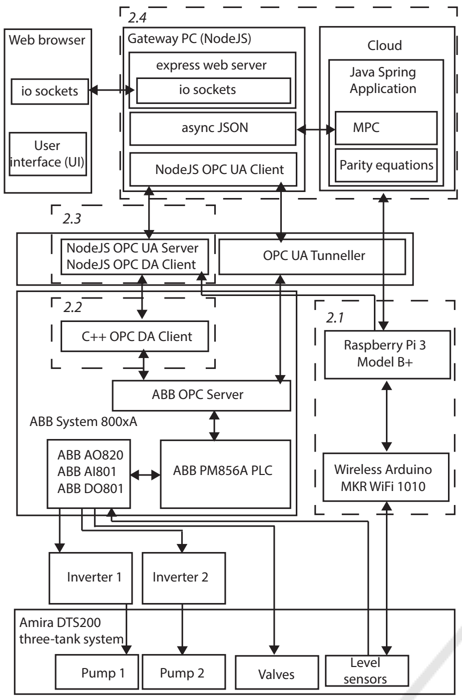

Figure 1: The architecture of the model predictive control and fault detection system in the cloud.

2.2 Optimized OPC DA Client

In this paper an OPC DA client has been implemented and it has been compiled with Visual Studio 2019 so that it would be available on the latest Windows operating system such as 10 and 11 and, for example, the latest NodeJS. Corresponding branches in the target OPC DA server must be found and listed. In an optimized way, the branch should be interpreted only when it is opened.

1. First, the position is moved to the root with the command

   ```
   ChangeBrowsePosition(OPC_BROWSE_TO,
   W2OLE(const_cast<wchar_t *>(
   parentItemPath.c_str() ) ) );
   ```

2. With the method

   ```
   ChangeBrowsePosition(OPC_BROWSE_DOWN,
   itemPath)
   ```

   we go down the branch.

3. All branch leaves are read with

   ```
   BrowseOPcitepmIDs(OPC_BRANCH, emptyString,
   VT_EMPTY, 0, &iEnum );
   ```

   and the name of the list with

   ```
   GetItemID(itemName, &FullName);
   ```

4. The item is first added to the group of type `IOPcitepmMgt` with the command

   ```
   AddItems(1, &itemDef, &pResults, &pErrors);
   ```

5. After that, the value can be written with the `IOPCSyncIO` object with the command

   ```
   Write(1, &hServer, &value, &pErrors)
   ```

   and read with command

   ```
   Read(OPC_DS_DEVICE, 1, &hServer,
   &pItemState, &pErrors)
   ```

2.3 NodeJS OPC DA Client

With NAN module of NodeJS you can use a template to wrap C++ functions and data structures within JavaScript objects so that they can be manipulated from JavaScript. In this document, NodeJS calls to the C++ OPC DA Client were implemented, so that branches of the OPC DA server can be called through the NodeJS OPC UA server in an optimized way.

1. `NAN_MODULE_INIT()` sets entry points for Node add-ons. It makes an object that corresponds to JavaScript's export command object.

2.
   ```
   Nan::Set(target, New<string>( \\
   "ReadOPCDAValue").ToLocalChecked(), \\
   GetFunction(New<functionTemplate>( \\
   ReadOPCDAValue)).ToLocalChecked());
   ```

   add a NodeJS method to read the OPC DA protocol value.

2.4 Gateway PC + Spring Framework

1. The gateway PC reads the data from the OPC UA server and sends the data so that the header contains the names of the data columns and the data as a whole is a table. Compared to name-value pairs, the data takes up significantly less space and can be created directly as a data table for the native language as shown in Fig. 2 and Fig. 3.
2. Spring framework automatically converts json-string into Java objects.

3 MODELLING OF THE THREE-TANK SYSTEM

The development and validation of the three-tank system are detailed in (Kortela, 2022). The three-tank system consists of tanks T_1, T_3, and T_2 with the same cross-sectional area A_b, as shown in Fig. 4. These cylindrical tanks are connected in series by a cylindrical pipe with cross-sectional area A_c. Liquid is collected in a reservoir and is pumped back into tanks T_1 and T_2 using pumps 1 and 2 to maintain their levels.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

```
let dataObject = {
    cname: "RaspberryPi"
    seconds: "Time (s)",
    ts_utc_tz: "Timestamp (UTC + TZ)",
    ts_local: "Timestamp (local)",
    staticdata: [{
            seconds: "0.124064",
            ts_utc_tz: "2023-11-01T08:01:59+02:00",
            ts_local: "2023-11-01 10:01:59.122155"
        },
        {
            seconds: "0.124064",
            ts_utc_tz: "2023-11-01T08:01:59+02:00",
            ts_local: "2023-11-01 10:01:59.122155"
        }],
    doubledata: [[0.1,0.2,0.3],
                 [0.4,0.5,0.5]],
    labelsdouble: [{key: "temperature"},
            {key: "humidity"},
            {key: "pressure"}],
    descriptionsdouble: [{key: "temperature PT-100"},
            {key: "%"},
            {key: "mbar"}]
    };
```

Figure 2: Optimized data structure to save data.

```
let dataObject2 = {
                cname: "raspberrypi",
                ts_local_start: "2021-11-01 10:01:59.122155",
                ts_local_end: "2025-11-01 10:01:59.122155"

            };
```

Figure 3: Optimized data structure to retrieve data.

All tanks are equipped with a piezo-resistive pressure transducer that measures the liquid level in the tank.

The flow rates of pumps 1 and 2 are represented by Q_1 and Q_2, respectively. The flow rate provided by a pump is proportional to the DC voltage applied to its motor. In the tested system, valves V_{13}, V_{32}, and V_{3O} were open and leakage valves V_{L1}, V_{L3}, and V_{L2} were closed. The linearized state-space model parameters are given by

$$A=\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}, B=\begin{bmatrix} \frac{1}{A_b} & 0 \\ 0 & \frac{1}{A_b} \\ 0 & 0 \end{bmatrix} \qquad (1)$$

$$C=\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, D=0$$

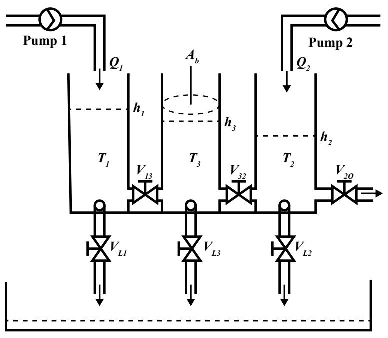

Figure 4: Three-tank system.

$$\begin{aligned}
a_{11} &= -\frac{\alpha_{13}A_c\sqrt{2g}}{2A_b\sqrt{h_{1s}-h_{3s}}}, a_{12}=a_{21}=0,\\
a_{13} &= a_{31} = \frac{\alpha_{13}A_c\sqrt{2g}}{2A_b\sqrt{h_{1s}-h_{3s}}},\\
a_{22} &= -\left(\frac{\alpha_{32}A_c\sqrt{2g}}{2A_b\sqrt{h_{3s}-h_{2s}}}+\frac{\alpha_{2O}A_c\sqrt{2g}}{2A_b\sqrt{h_{2s}}}\right),\\
a_{23} &= a_{32} = \frac{\alpha_{32}A_c\sqrt{2g}}{2A_b\sqrt{h_{3s}-h_{2s}}}\\
a_{33} &= -\left(\frac{\alpha_{13}A_c\sqrt{2g}}{2A_b\sqrt{h_{1s}-h_{3s}}}+\frac{\alpha_{32}A_c\sqrt{2g}}{2A_b\sqrt{h_{3s}-h_{2s}}}\right)
\end{aligned} \qquad (2)$$

where A is the state matrix, B is the input matrix, C is the output matrix, D is the matrix that describes which inputs affect directly the outputs, $\alpha_{ij} \in [0,1]$ denotes the outflow coefficient between tank i, j and out from the tank 2, g is the gravity constant, and h_{1s}, h_{2s} and h_{3s} are the operating points of the three levels, respectively.

4 MODEL PREDICTIVE CONTROL FOR THE THREE TANK PILOT SYSTEM

4.1 State-Space Model Based MPC

As a detailed physical model of the three-tank system was available, it was a natural choice to use the linearized version of that model directly with MPC. The inputs to the MPC are the reference values for the two water levels (r) and the measured process outputs for the levels (y). The outputs of the MPC are the manipulated variables, the speeds of the two water pumps (u). The linear state space system for the MPC is as follows (Maciejowski, 2002):

$$\begin{aligned} x(k+1) &= Ax(k)+Bu(k)+Ed(k)\\ z(k) &= Cx(k) \end{aligned} \qquad (3)$$

where x are the states, E is the disturbance matrix and d are the disturbances. The regularized l_2 output tracking problem with the input, the input rate of movement, and the output constraints is formulated as

$$\begin{aligned}
\min\phi &= \tfrac{1}{2}\textstyle\sum_{k=1}^{N_p} \|z(k)-r(k)\|^2_{Q_z}+\\
&\quad \tfrac{1}{2}\textstyle\sum_{k=1}^{N_p-1} \|\Delta u(k)\|^2_S\\
s.t.x(k+1) &= Ax(k)+Bu(k)+Ed(k),\\
&\quad k=0,1,\ldots,N_p-1\\
z(k) &= Cx(k), k=0,1,\ldots,N_p\\
u_{\min} &\le u(k) \le u_{\max}, k=0,1,\ldots,N_p-1\\
\Delta u_{\min} &\le \Delta u(k) \le \Delta u_{\max}, k=0,1,\ldots,N_p-1\\
z_{\min} &\le z(k) \le z_{\max}, k=1,2,\ldots,N_p
\end{aligned} \qquad (4)$$

where $\Delta u(k)=u(k)-u(k-1)$ and N_p is the prediction horizon, r is the future target vector, Q_z is the tracking error weight matrix, and S is the move suppression factor weight matrix. Sizes of the stacked matrices Z, R, U and D depend on the prediction horizon N_p.

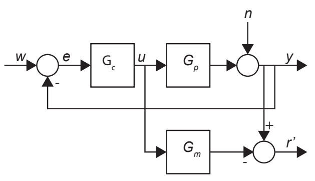

Figure 5: Fault detection of closed loop with parity equations and output error r'.

5 FAULT DETECTION WITH PARITY EQUATIONS

Parity equations were also chosen to identify the fault because the exact model of the three-tank system was available (Isermann, 2006).

5.1 Model-Based Methods for Closed-Loop Supervision

Application of parity equations in closed-loop is considered. As shown in Fig. 5 a residual r is generated by using a fixed model. The calculation of the output error is as follows:

$$r'(s) = y_p(s) - y_m(s) = y_p(s) - G_m(s)u(s) \quad (5)$$

$$y_p(s) = G_p(s)u(s) + n(s) \quad (6)$$

$$u(s) = \frac{G_c(s)}{1 + G_c(s)G_p(s)}(w(s) - n(s)) \quad (7)$$

$$\begin{aligned} r'(s) &= (G_p(s) - G_m(s))u(s) + n(s) \\ &= (G_p(s) - G_m(s))\frac{G_c(s)}{1+G_c(s)G_p(s)}(w(s) - n(s)) \\ &\quad + n(s) \\ &= \frac{G_c(s)(G_p(s) - G_m(s))}{1+G_c(s)G_p(s)}w(s) \\ &\quad + \frac{1+G_c(s)G_m(s)}{1+G_c(s)G_p(s)}n(s) \end{aligned} \quad (8)$$

If the model agree with the real process, $G_p(s) = G_(s)$, it holds

$$r'(s) = n(s) \quad (9)$$

6 THE ARCHITECTURE OF THE MODEL PREDICTIVE CONTROL FOR THE THREE TANK PILOT SYSTEM IN THE CLOUD

The experimental setup consists of a cloud with a Java Spring Application and MPC and the fault detection system implemented in Java. Apache Commons Math 3.6.1 API and oj! Algorithms (ojAlgo) were utilized to implement MPC, as shown in Fig. 1. The gateway PC reads the level measurements and writes the values of the pumps via the NodeJS OPC UA Server and alternatively through the OPC UA tunneller software. It communicates with the cloud through the JSON protocol. The web browser receives the new pump and level values through the IO socket and draws them to the user interface using HTML 5 canvas. The connections of the ABB PM856A PLC and cards are defined in ABB Control Builder M Professional.

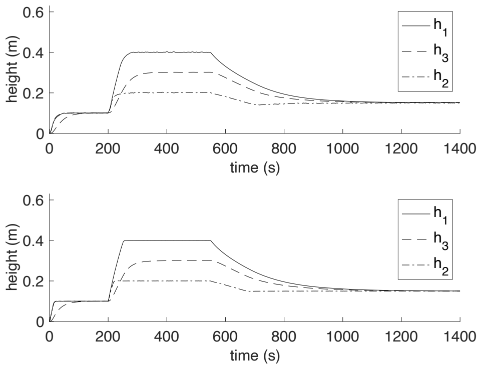

Figure 6: Levels' response in experimental setup utilizing OPC UA tunneller (above). Levels' response in experimental setup utilizing NodeJS OPC UA server and OPC DA client (below).

7 EXPERIMENTAL RESULTS OF THE OPTIMIZED SIGNAL PATH

Table 1: Three tank system parameters.

Cross section (tanks) (A_b)	0.0154m^2
Cross section (pipes) (A_c)	$5 \cdot 10^{-5}\text{m}^2$
Valve opening (α_{ij})	$\alpha_{ij} = 0.84$
Max flow rate (Q_{max})	$1.2 \cdot 10^{-4}\text{m}^3/\text{s}$
Maximum level (h_{max})	0.63m
Operating point	$Q_1 = 7 \cdot 10^{-5}\text{m}^3/\text{s}$
	$Q_2 = 4 \cdot 10^{-5}\text{m}^3/\text{s}$
	$h_1 = 0.45\text{m}$
	$h_2 = 0.25\text{m}$
	$h_3 = 0.35\text{m}$

7.1 Experimental Results of MPC of Three-Tank System

The MPC was tested on Heroku Cloud with ps:scale web=1, on a cloud server with an Intel Xeon Gold 6248 CPU and 16 GB of memory, as well as on a Macbook Pro with an Apple M1 processor and 16 GB of memory. The execution time of MPC on these

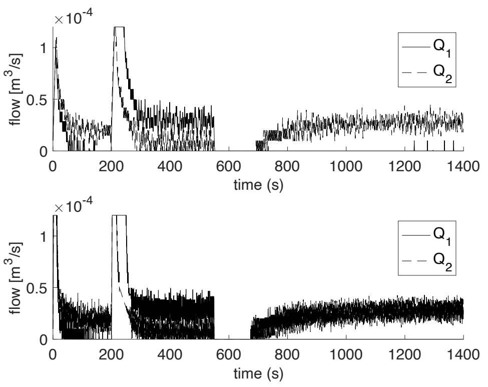

Figure 7: Input flow rates in experimental setup utilizing NodeJS OPC UA server and OPC DA client (above). Input flow rates in experimental setup utilizing OPC UA tunneller. (below).

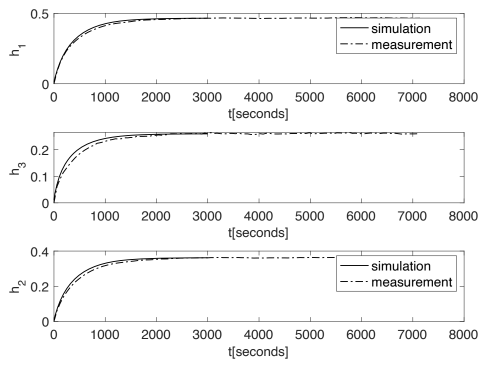

Figure 8: Accurate calibration of three tank system.

systems was approximately 100 ms, which was the limiting factor for the speed of MPC. The OPC UA tunneller software and ABB OPC Server with their 3 level variables and 2 pump variables limited the control interval to around 0.5 second. The parameters were substituted by The MPC was discretized with a sampling interval of 200 milliseconds when using the NodeJS OPC UA and OPC DA path, and a sampling interval of 1 second when using the OPC UA tunneller path. The model used the parameters defined in Table 1. Figures 6 and 7 show the response of the three tank levels and input flow rates using the NodeJS OPC UA and OPC DA path and the path with OPC UA tunneller. Due to its smaller control interval of 200 milliseconds, the former controller provides a faster response.

7.2 Experimental Results of Fault Detection and Diagnosis System

The testing of the fault detection and diagnosis system was conducted using the same system as for testing MPC, additionally including a Wireless Arduino MK WiFi 1010 device and a Raspberry Pi 3 Model B+ device. The mA data for the surface heights h_1, h_3 and h_2 were calibrated so that they showed the correct millimeter value at the top and bottom of the tank with an accuracy of about a millimeter. Pump 1 and Pump 2 were calibrated with 1% percent interval to obtain an exact calibrated equation for the pumps. This was done for both pumps. In addition, with the help of the collected data, the α_{ij} parameters were accurately identified. The validation between digital twin and data is shown in Fig. 8. The parameters are show in Table 2. In the first test, the delay was measured by measuring how long it takes to read and write the measurement value to the cloud. Test results show that the delay is 409 milliseconds when using OPC UA tunneller software. When reading data utilizing the Arduino MK WiFi 1010, the delay is 36 milliseconds. The delay is only 22 milliseconds when the Gateway pc reads data through the C++ OPC DA and NodeJS combination. In the second test, the signal path is tested with the three-tank system with the parity equations with transfer functions implemented in the cloud. The Gateway PC read the values utilizing a combination of C++ OPC DA and NodeJS and wrote them to the cloud service. The set point values of 0.4 and 0.2 were set for level 1 and level 2, respectively. Test results show that the digital twin detects a fault in 56 seconds. To conclude, OPC DA no longer meets current safety standards, however by combining it with OPC UA, an optimized and safe signal path even for the latest Windows version is achieved.

8 CONCLUSIONS

OPC DA has not been updated earlier to function with the latest programming libraries and operating systems, which significantly reduced the performance of automation systems. The developed optimized signal path using C++ OPC DA and NodeJS was compared

Table 2: Accurate model of the faulty pumps.

Variable	Value
$U_{Signal1}$	$0-100$
$U_{Signal2}$	$0-100$
a_1	-32.41
b_1	1.475
a_2	-33.34
b_2	1.440
$U_{Pump}1$	$a_1 + b_1 * U_{Signal1}$
$U_{Pump}2$	$a_2 + b_2 * U_{Signal2}$
$flow1$	$1.0 * 10e - 6 * U_{Pump1} * 0.1$
$flow2$	$1.0 * 10e - 6 * U_{Pump2} * 0.1$

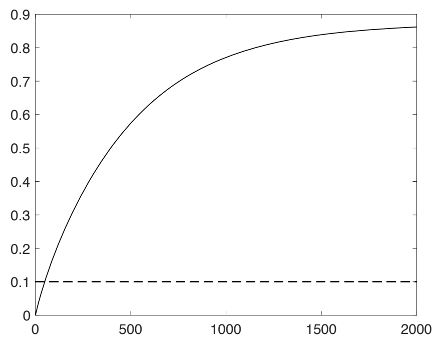

Figure 9: Fault in the pump 1 and pump 2 in the closed loop three-tank system.

to the OPC UA tunneler implementation through experiments conducted on a real three-tank pilot system. Due to the optimized signal path, the control interval of MPC is 5 times smaller, resulting in a faster response from the controller. In fault diagnosis and detection, the delay is only 22 milliseconds compared to 408 milliseconds when using OPC UA tunneller software. Test results show that the fault detection system detects a fault in 56 seconds.

ACKNOWLEDGEMENTS

Pumps calibration was implemented by Otacilio Bezerra Leite Neto.

This work has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No. 101057083 – Zero-SWARM.

REFERENCES

Alhassan Osman, T. K. and Arıcı, M. (2023). Robust adaptive control of a quadruple tank process with sliding mode and pole placement control strategies. *IETE Journal of Research*, 69(5):2412–2425.

Bonomi, F., Milito, R., Zhu, J., and Addepalli, S. (2012). Fog computing and its role in the internet of things. In *MCC '12: Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16.

Denzler, P., Ashjaei, M., Frühwirth, T., Ebirim, V. N., and Kastner, W. (2022). Concurrent opc ua information model access, enabling real-time opc ua pubsub. In *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*.

DIN (2016). Din spec 91345:2016-04, reference architecture model industrie 4.0 (rami4.0). Technical report.

European Commission (2017). Digital transformation monitor. Technical report. Accessed 5 December 2022.

Evensen, L. (2013). Why a reference architecture is important for you. Technical report. Accessed 5 December 2022.

Ghazali, M. R., A. M. A. . R. I. R. M. T. (2019). Adaptive safe experimentation dynamics for data-driven neuroendocrine-pid control of mimo systems. *IETE Journal of Research*, 68(3):1611–1624.

Govind, K. R. A., Mahapatra, S., and Mahapatro, S. R. (2023). Design of an optimal control strategy for coupled tank systems using nonlinear constraint optimization with kharitonov-hurwitz stability analysis. *IEEE Access*, 11:72618–72629.

Großmann, D., Bregulla, M., Banerjee, S., Schulz, D., and Braun, R. (2014). Opc ua server aggregation - the foundation for an internet of portals. In *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*.

IEC (2015). Factory of the future. Technical report. Accessed 5 December 2022.

IEEE (2022). Systems and software engineering – architecture description. Technical report. Accessed 5 December 2022.

Industrial Internet Consortium (2019). The industrial internet of things volume g1: Reference architecture. Technical report.

Isermann, R. (2006). *Fault-Diagnosis Systems, An Introduction from Fault Detection to Fault Tolerance*. Springer, Berlin.

Kortela, J. (2022). Model-predictive control for the three-tank system utilizing an industrial automation system. *ACS Omega*, 7(22):18605–18611.

Maciejowski, J. M. (2002). *Predictive Control with Constraints, 1st ed.* Prentice Hal, Harlow.

Müller, A., Schnieders, T., Storms, S., and Herfs, W. (2022). Integration method of custom information models into existing opc ua servers. In *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*.

Reiswich, E. and Fay, A. (2012). Strategy for the amendment of plant information models by means of opc ua. In *IEEE 10th International Conference on Industrial Informatics*.

Samuel Emebu, Marek Kubalčík, C. J. B. D. J. (2023). A comparative study of linear and nonlinear optimal control of a three-tank system. *ISA Transactions*, 132:419–427.

Ujjwal Manikya Nath, C. D. and Mudi, R. K. (2023). Review on imc-based pid controller design approach with experimental validations. *IETE Journal of Research*, 69(3):1640–1660.

VDI Verein Deutscher Ingenieure e.V. (2015). Reference architecture model industrie 4.0 (rami4.0). Technical report. Accessed 5 December 2022.