Contract-Based Runtime Monitoring for a Smart House Digital Twin

Muhammad Naeem, Cristina Seceleanu, Tiberiu Seceleanu
Mälardalen University, Västerås, Sweden; email: {muhammad.naeem, cristina.seceleanu, tiberiu.seceleanu}@mdu.se

Abstract

Digital Twins (DTs) are increasingly used to support monitoring, prediction, and decision-making in cyber-physical systems. However, ensuring the reliability of DT-based systems remains challenging, particularly when data-driven models are used for control and prediction. The D-RODS project addresses this challenge by developing a framework for dynamic and robust distributed DTs that integrates artificial intelligence with verification and validation techniques. In this work-in-progress paper, we present a prototype that applies the D-RODS framework to a smart house scenario. The system maintains the temperature of a physical environment within recommended limits, while enabling monitoring and automated control through a neural network DT implemented in MATLAB, and connected to the physical system through a supervisory application. The application incorporates a contract-based observer that verifies input signals and controller decisions at runtime. When violations are detected, control is switched from the physical controller to the DT controller. Initial experiments show that the DT controller closely follows the behavior of the physical controller, while the temperature prediction model exhibits deviations in some scenarios.

Keywords: Digital Twins, Cyber-Physical Systems, Runtime Monitoring, Contract-Based Verification

1 Introduction

Digital Twins (DTs) are increasingly used to support monitoring, prediction, and control in cyber-physical systems. A DT provides a virtual representation of a physical system that interacts with it using real-time data. As DTs become integrated into operational environments, ensuring reliable synchronization between the physical system and its digital counterpart becomes essential. In practice, discrepancies may arise due to sensor faults, communication delays, environmental disturbances, or model inaccuracies, highlighting the need for runtime monitoring and reliability mechanisms.

The D-RODS (Digital Twin Framework for Dynamic and Robust Distributed Systems)[1] project aims to address these challenges by developing a framework for building reliable and adaptive DT systems. The framework combines artificial intelligence with verification and validation techniques to support trustworthy DT-based applications. Previous studies within the project have investigated contract-based verification of DTs, self-adaptation mechanisms for distributed DT systems, and service-oriented architectures for integrating DT components [1–5]. Related work in the broader DT literature has demonstrated the potential of DTs for monitoring, optimization, and decision support in cyber-physical systems across several domains [6–14].

In this paper, we present a prototype that applies the D-RODS framework to a smart house scenario. The goal of the system is to maintain the temperature of a physical environment within recommended limits while enabling monitoring and automated control through a DT. The DT is implemented in MATLAB and connected to the physical prototype through a supervisory application that manages communication, monitoring, and control. The application incorporates a runtime observer that verifies sensor inputs and controller decisions using predefined contracts. Observer rule: if assumptions fail, activate input DT; if guarantees fail, activate controller DT.

Although DTs have been widely explored for monitoring and optimization tasks, ensuring reliable DT behavior during runtime remains a challenge. Many existing implementations focus on predictive modeling and data integration but provide limited mechanisms for detecting inconsistencies between the physical system and the digital model.

Beyond the smart-house use case, the main contribution is the practical integration of runtime contract monitoring, fault detection, and automatic DT fallback control in a live cyber-physical prototype. The main contributions of this work are: (i) the development of a smart house DT prototype as an experimental platform for evaluating the D-RODS framework, (ii) the integration of a MATLAB-based application that connects the DT with the physical system for real-time monitoring and control, (iii) the implementation of a contract-based runtime observer that verifies both input signals and controller decisions, and (iv) a mechanism that switches control to the DT when violations are detected.

Initial observations show that the DT controller closely follows the behavior of the physical controller, while the temperature prediction model exhibits deviations in some scenarios, highlighting challenges related to model accuracy and synchronization between the digital and physical systems.

[1] https://www.es.mdu.se/projects/615-D_RODS

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 presents the smart house use case, MATLAB application and observer mechanism. Section 4 describes the DT models. Section 5 introduces the contract model for runtime verification. Section 6 discusses initial results, and Section 7 concludes the paper.

2 Related Work

DT technology has been widely studied for monitoring and controlling cyber-physical systems. A key challenge in DT-based systems is maintaining synchronization between the physical system and its digital counterpart. Zhang et al. emphasize the importance of continuously updating DT models to preserve consistency between the physical and virtual entities during system operation [6].

DTs have been applied in several industrial domains, particularly in manufacturing systems. Yuan et al. proposed a DT-driven optimization method for electrical cable production lines using real-time data to improve operational efficiency [7]. Similarly, Leng et al. developed a DT platform for reconfigurable manufacturing systems that adapts the digital model to changes in the physical system [8]. Other studies have explored integrating DTs with artificial intelligence and edge computing to support real-time decision making and system monitoring [9]. Structured modeling approaches have also been proposed to improve the representation of complex physical systems [10].

DT applications have also been explored in areas such as building systems, energy infrastructure, and healthcare. Examples include DT-based cyber-physical systems for building heating systems [11], virtual testbeds for power system DT applications [12], and DT frameworks for smart hospitals that integrate heterogeneous data sources [13].

While these studies demonstrate the benefits of DTs for monitoring and optimization, ensuring reliable DT operation during runtime remains an open challenge. Our previous work on contract-based verification of DTs introduced a formal approach for specifying and verifying system behavior [1]. In contrast to prior DT studies focused on monitoring or optimization, this work emphasizes runtime contract verification and fault-triggered control fallback.

3 Use Case and Method

To evaluate the proposed approach, we developed a small smart house prototype connected to a DT. The system serves as a controlled environment for studying the interaction between a physical system and its digital counterpart within the D-RODS framework.

The physical system is implemented using an ESP32 development module connected to a DHT11 temperature sensor and three actuators: a DC fan, a warning LED, and a buzzer. The sensor continuously measures the ambient temperature and sends the data to the supervisory application through serial communication. Based on the measured temperature, the actuators are activated to regulate the environment and signal abnormal operating conditions.

A Neural-Network-based Digital Twin (NNDT) of the system is implemented in MATLAB and integrated with the physical prototype through a supervisory application. The application manages communication with the hardware, visualizes system behavior, and monitors the states of the actuators. It also allows controlled disturbances to be introduced, such as disabling the temperature input or modifying controller outputs, enabling the evaluation of abnormal operating scenarios.

To improve reliability, the application incorporates a runtime observer based on the contract-based verification approach described in Section 5. The observer monitors both sensor inputs and controller decisions and evaluates them against predefined contracts. When a contract violation is detected, the system switches control from the physical controller to the corresponding DT model. In particular, input-related violations activate the temperature DT, while controller-related violations activate the controller DT. Once no contract violations are detected for a predefined observation window, control can be restored to the physical system. Switching is event-triggered by contract violations and remains active until stable contract satisfaction is re-established for a predefined interval.

Figure 1 illustrates the overall architecture of the prototype, including the physical smart house, the MATLAB supervisory application, and the DT models.

4 Digital Twin (DT) Models

The DT of the smart house system consists of two types of models: a temperature prediction model and controller models that replicate the behavior of the physical controller. These models operate together to reproduce the behavior of the physical system and allow the DT to take control when required. The neural-network models were trained offline in MATLAB using data collected from the physical system during normal operation. The temperature DT uses a feedforward NN with lagged temperature inputs, while actuator DTs use binary classifiers trained from temperature-action traces.

Temperature DT: The temperature DT predicts the short-term evolution of the indoor temperature based on recent temperature measurements. Instead of directly mapping the current temperature to itself, the model uses a short history window of temperature values to estimate the next temperature sample. This approach allows the DT to capture short-term temperature trends.

The prediction model is implemented using a feedforward NN trained to perform one-step-ahead forecasting [15]. The model uses the three most recent temperature measurements as input and predicts the next temperature value:

$$\hat{T}(t+1) = f(T(t), T(t-1), T(t-2)), \qquad (1)$$

where $T(t)$ denotes the measured temperature at time t, $\hat{T}(t+1)$ represents the predicted temperature at the next time step, and $f(\cdot)$ is the nonlinear function learned by the NN during training. The current predictor uses only historical temperature values. Future versions will also include actuator states (e.g., fan ON/OFF), which influence thermal dynamics.

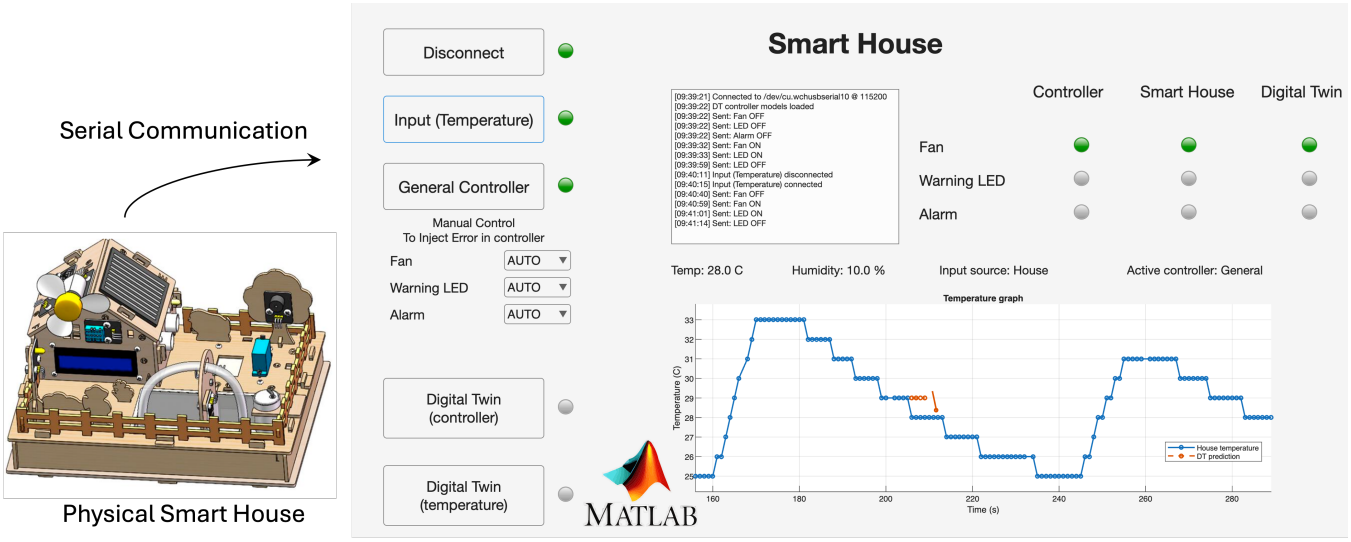

Figure 1: Overview of the smart house prototype, MATLAB supervisory application, and DT integration used for runtime monitoring and control.

During runtime, the DT continuously computes predicted temperature values using the most recent measurements. Both the measured temperature and the DT-predicted temperature are monitored within the supervisory application. When the physical temperature input becomes unavailable, disabled, or unreliable, the DT-predicted temperature signal is used as a substitute input for the control system, enabling the system to continue operating.

Controller DT: The controller DT replicates the behavior of the physical controller that determines the states of the fan, warning LED, and alarm. These actuators operate in binary states and are therefore modeled using binary neural network (BNN) classifiers [16]. Each actuator model was trained independently as a binary classifier using sigmoid output activation and binary cross-entropy loss.

To capture the temporal behavior of the controller, the neural network considers a sequence of past temperature values as input. This lagged input structure allows the model to represent delayed system responses and temporal dependencies in the control decisions. The controller model can be represented as

$$\tilde{c}(t) = g(T(t), T(t-1), \ldots, T(t-k)) \qquad (2)$$

where $\tilde{c}(t)$ represents the predicted control action at time t, $T(t)$ is the temperature input, and k denotes the number of previous time steps considered by the model. The function $g(\cdot)$ is learned from the training data collected from the physical system.

Separate models are trained for each actuator to predict their binary states: $\tilde{f}(t) = g_f(T(t), \ldots, T(t-k))$, $\tilde{l}(t) = g_l(T(t), \ldots, T(t-k))$, and $\tilde{a}(t) = g_a(T(t), \ldots, T(t-k))$, where $\tilde{f}(t)$, $\tilde{l}(t)$, and $\tilde{a}(t)$ represent the predicted states of the fan, warning LED, and alarm, respectively.

These controller models enable the DT to reproduce the decision behavior of the physical controller. When the runtime observer detects violations in the physical system, the DT models are used to generate control actions and maintain system operation.

5 Contract Model

To ensure reliable system operation, the proposed framework applies *contract-based* verification to monitor the behavior of both the physical system and the DT during runtime. Contracts formally describe the expected behavior of system components by specifying assumptions about their inputs and guarantees about their outputs. The contract structure follows the contract-based verification framework for DTs introduced in our previous work [1].

Definition 1 (Contract). A contract C is defined as a pair (A, G) where A represents the assumptions on the inputs of a component and G represents the guarantees that must hold for its outputs whenever the assumptions are satisfied.

Contracts are also used to reason about the behavior of NNDT. Hence, contracts are evaluated for both the physical system and the DT outputs to ensure consistency during runtime. Since neural network models are data-driven, small deviations between predicted and observed values may occur. Therefore, instead of strict equality, the contracts consider the mean prediction error over a time window. Let $e(t) = y(t) - \hat{y}(t)$ denote the difference between the measured value $y(t)$ and the DT prediction $\hat{y}(t)$. The mean error over N samples is defined as

$$\hat{e}(t) = \frac{1}{N} \sum_{i=1}^{N} e(t-i).$$

where $\hat{e}(t)$ represents the stabilized mean output. These cross-cutting assumptions hold for all contracts. If the mean error remains within a predefined bound, the DT behavior is considered consistent with the physical system; otherwise, a contract violation may be detected.

The following contracts describe the expected behavior of the smart house controller and the validity of the temperature input signal.

C1 (Fan activation): $A : \forall t, \hat{T}(t) > 30,\ G : f(t)$

Contract ***C1*** states that when the temperature exceeds 30°C, the fan must be activated.

C2 (Fan deactivation): $A : \forall t, f(t-1) \wedge \hat{T}(t) < 25,\ G : \neg f(t)$

Contract ***C2*** ensures that if the fan is running and the temperature drops below 25°C, the fan should be turned off.

C3 (Warning LED behavior): $A : \ \hat{T}(t) > 30 \vee \hat{T}(t) < 25, G : l(t)$

The warning LED must be activated when the temperature leaves the normal operating range, which is what contract ***C3*** models.

C4 (Alarm activation): $A : \ \forall t,\ \hat{T}(t) > 35 \vee \hat{T}(t) < 20,\ G : a(t)$

The alarm buzzer must be activated when the temperature reaches a critical level (by contract ***C4***).

C5 (Temperature consistency): $A : \ \forall t,\ |\hat{T}(t) - \hat{T}(t-1)| \leq \Delta_T$

This contract ensures that the temperature signal varies smoothly over time and prevents unrealistic, sudden changes in the measured temperature.

During runtime, the observer continuously evaluates these contracts using the incoming sensor data and controller outputs. If an input assumption is violated, the observer identifies abnormal sensor behavior and activates the temperature DT to generate predicted inputs. If a controller guarantee is violated, the observer detects incorrect controller behavior and switches control to the DT controller. Control can return to the physical system once the contract conditions are satisfied again. Current contracts use threshold-based rules for clarity and rapid prototyping. Future work will extend them with temporal and multi-component contracts.

6 Results and Discussions

The performance of the proposed DT framework was evaluated using experimental data from the smart house prototype. The evaluation focuses on the DT controller's ability to reproduce the physical control behavior and the DT temperature model's capability to provide substitute inputs when sensor data becomes unavailable. As a baseline, system behavior without DT fallback leads to loss of control during missing sensor input, whereas the proposed framework maintains operation using DT estimates.

Fig. 2 compares the fan control signal generated by the physical controller and the DT controller. In the initial implementation (Fig. 2(a)), a delay is observed in the DT-predicted signal. This behavior is similar to observations reported in our previous work on service-oriented DT frameworks. To address this issue, a lag was introduced in the temperature input used by the DT controller. As shown in Fig. 2(b), incorporating the

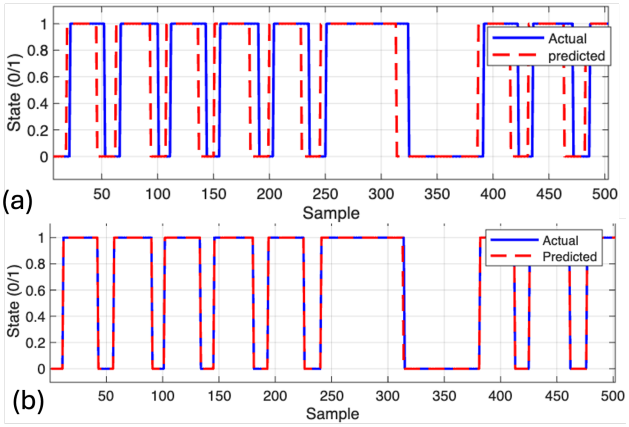

Figure 2: Comparison of physical and DT fan-control outputs before and after lag compensation. (a) Initial DT prediction showing delay. (b) Improved prediction after introducing lagged temperature input.

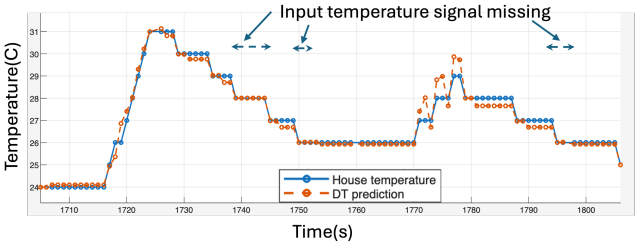

Figure 3: Measured and DT-predicted temperature during normal and simulated sensor-fault intervals.

lagged input significantly improves the alignment between the predicted and actual control signals. In fault scenarios, contract violations were detected within one sampling cycle, after which DT control was activated.

Fig. 3 shows the comparison between the measured house temperature and the DT-predicted temperature obtained from the supervisory application. The predicted signal generally follows the trend of the actual temperature, with small deviations occurring at certain points. To evaluate the robustness of the DT model, sections of the temperature input were intentionally removed to simulate sensor faults. During these periods, the DT continues to generate temperature estimates based on previous observations, allowing the system to maintain operation. Prediction deviations and transient spikes are mainly due to sensor noise, limited training data, and omission of fan-control input. In safety-critical settings, DT outputs can be filtered before substitution. Although variations in the temperature signal can introduce minor prediction errors, the DT model preserves the overall system behavior and provides a reliable substitute signal when the physical input becomes unavailable.

This paper presents an initial prototype; therefore, the current evaluation focuses on behavioral validation. A more extensive quantitative evaluation of prediction accuracy, detection latency, and robustness is planned as future work.

7 Conclusion and Future Work

This paper presented an initial prototype of a DT system for a smart house use case within the D-RODS framework. The system integrates a physical smart house prototype with DT models implemented in MATLAB and a supervisory application that enables real-time communication, monitoring, and control. A contract-based runtime observer was introduced to verify sensor inputs and controller decisions during operation. When contract violations are detected, the system can switch control to the corresponding DT model, allowing the system to continue operating under abnormal conditions. Initial experiments indicate that the DT controller is able to reproduce the control behavior of the physical controller with high similarity. However, deviations were observed in the temperature prediction model in some scenarios, highlighting challenges related to model accuracy and synchronization between the physical and digital systems. These results indicate the feasibility of contract-guided DT fallback for reliable CPS operation

Although variations in the temperature signal can introduce minor prediction errors, the DT model preserves the overall system behavior and provides a reliable substitute signal when the physical input becomes unavailable. Current limitations include simplified contracts, small-scale experiments, and limited training data. As future work, we plan to refine the DT models to improve prediction accuracy and extend the smart house prototype with additional operational scenarios and a large dataset. Another direction is to integrate richer timed-automata-based contracts from our earlier work for stronger temporal guarantees.

Acknowledgements. The authors are partly supported by Vinnova's *Advanced digitalization* programme in the project D-RODS (ID: 2023-00244).

References

[1] M. Naeem and C. Seceleanu, "Contract-based verification of digital twins," in *International Conference on Engineering of Complex Computer Systems*, pp. 338–357, Springer, 2025.

[2] A. Cuzzocrea, C. Seceleanu, and T. Seceleanu, "A self-adaptation framework for supporting distributed computing based on industry-scale digital twins," in *2025 IEEE International Conference on Big Data (BigData)*, pp. 4504–4510, IEEE, 2025.

[3] R. Gu, T. Seceleanu, N. Xiong, and M. Naeem, "A service-oriented digital twin framework for dynamic and robust distributed systems," in *2024 IEEE International Conference on Software Services Engineering (SSE)*, pp. 66–73, IEEE, 2024.

[4] T. Seceleanu, N. Xiong, E. P. Enoiu, and C. Seceleanu, "Building a digital twin framework for dynamic and robust distributed systems," in *International Conference on Engineering of Computer-Based Systems*, pp. 254–258, Springer, 2023.

[5] M. Naeem, C. Seceleanu, A. Isaksson, and T. Seceleanu, "Efficient multi-level mine dewatering using uppaal stratego," in *Proceedings of the 28th International Symposium on Formal Methods (FM 2026) - Industry Day*, Springer, 2026.

[6] B. Zhang, G. Ding, Q. Zheng, K. Zhang, and S. Qin, "Iterative updating of digital twin for equipment: Progress, challenges, and trends," *Advanced Engineering Informatics*, vol. 62, p. 102773, 2024.

[7] G. Yuan, X. Liu, C. Zhu, C. Wang, M. Zhu, and Y. Sun, "Multi-objective coupling optimization of electrical cable intelligent production line driven by digital twin," *Robotics and Computer-Integrated Manufacturing*, vol. 86, p. 102682, 2024.

[8] B. Leng, S. Gao, T. Xia, E. Pan, J. Seidelmann, H. Wang, and L. Xi, "Digital twin monitoring and simulation integrated platform for reconfigurable manufacturing systems," *Advanced Engineering Informatics*, vol. 58, p. 102141, 2023.

[9] X. Liu, T. Xie, B. Wu, M. Li, and H. Li, "Construction of digital twin workshop integrated with edge computing and deep learning," in *Journal of Physics: Conference Series*, vol. 2816, p. 012008, IOP Publishing, 2024.

[10] F. Pei, S. Chen, Q. Li, S. Guo, Y. Xi, and P. Huang, "Construction method of cnc machine tool digital twin model based on the four-layer framework," *IEEE Access*, 2025.

[11] O. Y. Maryasin, "Digital twin of building heating substation: An example of a digital twin of a cyber-physical system," in *Cyber-Physical Systems: Modelling and Industrial Application*, pp. 61–73, Springer, 2022.

[12] Z. Shen, F. Arraño-Vargas, and G. Konstantinou, "Virtual testbed for development and evaluation of power system digital twins and their applications," *Sustainable Energy, Grids and Networks*, vol. 38, p. 101331, 2024.

[13] Y. Han, Y. Li, Y. Li, B. Yang, and L. Cao, "Digital twinning for smart hospital operations: Framework and proof of concept," *Technology in Society*, vol. 74, p. 102317, 2023.

[14] M. H. Rahim, N. Javaid, M. B. Janjua, M. Naeem, Z. A. Khan, and U. Qasim, "Comparative assessment of performance for home energy management controller in smart grid," in *2016 International Conference on Intelligent Networking and Collaborative Systems (INCoS)*, pp. 244–250, IEEE, 2016.

[15] R. Eldan and O. Shamir, "The power of depth for feedforward neural networks," in *Conference on learning theory*, pp. 907–940, PMLR, 2016.

[16] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," *Advances in neural information processing systems*, vol. 29, 2016.