engineering proceedings

Proceeding Paper

Digital Twin of an Automated Production Line †

Vanya Georgieva * and Hristiyan Tsanev

Department of Automation of Discrete Manufacturing, Technical University of Sofia, 1756 Sofia, Bulgaria; hristian.tsanev@gmail.com

* Correspondence: vgeorgieva@tu-sofia.bg

† Presented at the International Conference on Electronics, Engineering Physics and Earth Science (EEPES2026), Bandirma, Turkey, 24–27 June 2026.

Abstract

The automotive and logistics industries rely on automation to ensure operational efficiency and maintain high throughput. However, a critical limitation remains in achieving real-time spatial monitoring of dynamic assets and complex workflows. This paper proposes a scalable, cloud-hosted Digital Twin architecture to bridge the gap between physical operations and digital oversight. By simulating high-frequency Internet of Things (IoT) telemetry data, the system replicates the exact states and behaviors of real-world physical assets. This continuous data stream is processed in the cloud and mapped onto an interactive, 3D web-based dashboard. The solution provides stakeholders with comprehensive situational awareness, offering a robust framework for improving spatial tracking, proactive decision-making, and continuous optimization in logistical environments.

Keywords: digital twin; IoT telemetry simulation; cloud architecture; real-time spatial monitoring; 3D web dashboard

1. Introduction

In recent years, the automotive and logistics sectors have shifted, increasingly relying on advanced automation to maximize efficiency, minimize errors, and maintain a competitive edge in a fast-paced global market [1]. While automation has streamlined many linear processes, managing complex, multi-dimensional operations requires deeper operational insight. A major challenge remains the lack of comprehensive, real-time spatial monitoring. Traditional sensor networks and tracking methods typically yield flat, isolated data points. These legacy systems fail to provide the granular, three-dimensional context needed to identify spatial bottlenecks, optimize facility layouts, or execute rapid, context-aware operational adjustments.

To overcome these limitations, implementing a Digital Twin—a dynamic, synchronized virtual representation of a physical system—has emerged as a transformative solution [2]. This paper details the development of a highly scalable, cloud-hosted Digital Twin designed to address monitoring deficits in automotive and logistical applications. Our project simulates continuous, high-frequency Internet of Things (IoT) telemetry data, accurately emulating the precise movements, states, and conditions of industrial assets. By processing this simulated telemetry within a robust cloud infrastructure, the data is mapped to a live, interactive 3D web dashboard. This spatial, visually driven approach enhances immediate situational awareness for operators and establishes a foundational tool for continuous workflow optimization, bridging physical execution and digital oversight.

Check for updates

Academic Editors: Teodor Iliev, Ivaylo Stoyanov, Grigor Mihaylov and Selahattin Kosunalp

Published: 10 September 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

Scientific Novelty and Contributions

While commercial cloud-based platforms provide the foundational data infrastructure, the scientific novelty of this research lies in the architectural methodology developed specifically for Automotive Climate Control manufacturing. Existing literature frequently treats digital twins either as purely theoretical models or as isolated kinematic simulations [3]. This paper introduces a novel, multi-domain semantic graph topology that unifies disparate industrial parameters—ranging from mechanical metrics (Fastening Station torque) to thermodynamic variables (Thermal Chamber testing)—into a single, cohesive Hub-and-Spoke data structure.

The primary scientific contributions of this work are threefold:

1. The formalization of standardized, highly specialized Digital Twins Definition Language (DTDL) (v2 interfaces) schemas optimized specifically for automotive End-of-Line Testing (EOLT) and assembly.
2. The development of a scalable edge-to-cloud architectural blueprint that bridges the semantic gap between raw machine telemetry and spatial operational intelligence.
3. The demonstration of a multi-variable data integration method that successfully maps complex quality control metrics directly onto a 3D interface, drastically reducing the deployment friction for human-in-the-loop monitoring systems in real-world manufacturing environments.

2. The Proposed Solution

To bridge the gap between physical execution and digital oversight, this project proposes implementing an Azure Digital Twins (ADT; Microsoft Corp., Redmont, WA, USA) architecture [1]. By creating a synchronized, data-driven virtual replica of the industrial environment, this ADT shifts the operational paradigm from a reactive physical system to a proactive digital one. Instead of relying on manual inspections, operators can use the digital twin to instantly visualize anomalies in their exact spatial context. When efficiency drops, the system's real-time data ingestion lets stakeholders remotely diagnose issues, predict potential mechanical failures, and optimize system performance virtually before any physical intervention.

Project Scope

- DTDL Modeling: The system's foundation relies on the Digital Twins Definition Language (DTDL) to accurately define the ontology, properties, and relationships of physical assets, ensuring a standardized, highly scalable digital topology.
- Data Simulation via REST API: To replicate live industrial conditions, custom PowerShell scripts generate synthetic, high-frequency IoT telemetry. This simulated data is continuously pushed to the ADT environment via REST API endpoints, mirroring the data stream of physical sensors.
- 3D Scene Visualization: The final pillar involves mapping the dynamic telemetry data from the ADT to an interactive 3D web dashboard. This provides an intuitive, spatial interface where operators can visually monitor the real-time states, movements, and health of the simulated assets.

3. System Architecture: The "Digital" Bill of Materials

In a traditional physical engineering project, the Bill of Materials (BOM) is an exhaustive inventory of all tangible hardware components—such as actuators, sensors, and structural frameworks—required for assembly [1]. However, when building a cloud-hosted virtual environment, this shifts to a "Digital" Bill of Materials. Instead of physical hardware, this digital inventory includes data models, scripting interfaces, telemetry pipelines, and visualization frameworks needed to architect the twin. This digital infrastructure is highly modular, allowing rapid deployment and iteration without physical manufacturing lead times.

https://doi.org/10.3390/engproc2026154079

3.1. Component Modeling with DTDL

The foundational layer of this digital architecture is built using the Digital Twins Definition Language (DTDL) [4]. DTDL is an open, JSON-LD v.1.1 (JavaScript Object Notation for Linked Data) based modeling language utilized to define the precise semantics, ontology, and operational characteristics of the physical assets being replicated.

3.2. Digital Bill of Materials

While a traditional CAD model describes the physical geometry of a machine, DTDL describes its data behavior. Using DTDL, every component within the logistical environment is systematically defined through specific elements:

- Properties: Static or slow-changing data points that define the asset's baseline state, such as manufacturer details, serial numbers, or maximum payload capacities.
- Telemetry: The definition of high-frequency data streams emitted by the asset's simulated IoT sensors, such as current operating temperature, motor RPM, or specific joint coordinates.
- Relationships: The topological definitions of how different digital entities interact with one another within the spatial graph (e.g., defining that a specific robotic arm is "located in" Zone A, or "feeds into" Conveyor Belt B).

By utilizing DTDL, the system establishes a standardized, extensible vocabulary. This ensures that the Azure Digital Twins environment can accurately parse, contextualize, and map the incoming simulated telemetry to the correct digital representations of the physical world [5]. The resulting digital bill of materials is summarized in Table 1.

Table 1. Digital BOM (DTDL Models).

Digital Component	DTDL Identifier (@id)	Telemetry/Properties (Schema)	System Function
Parcel Sorting System	dtmi:com:example: ParcelSortingSystem;1	System Status (string)	Acts as the central logic controller and parent node, maintaining management relationships with all peripheral hardware.
Conveyor Belt	dtmi:com:example: ConveyorBelt;1	BeltSpeed (double) IsOccupied (boolean)	Represents the transport layer. Receives simulated speed and occupancy data to monitor throughput.
Robotic Arm	dtmi:com:example: RoboticArm;1	ArmStatus (string) Efficiency (double)	The automated execution unit. Tracks operational phases (e.g., "Sorting Parcel") and performance metrics.
Inspection System	dtmi:com:example: InspectSystem;1	ScanRate (double)	The quality control node. Logically linked to the central hub to process sorting logic.
Camera	dtmi:com:example: Camera;1	IsActive (boolean)	The visual monitoring edge device, feeding data to the Inspection System.

3.3. JSON-LD Blueprint Example

To ensure the Azure cloud (Microsoft Corp., Redmond, WA, USA) could accurately parse incoming telemetry, each component requires a strict JSON-LD definition. Below is an excerpt of the custom DTDL blueprint engineered for the Robotic Arm node. This blueprint establishes the "buckets" (ArmStatus and Efficiency) that will later be populated by the PowerShell simulation (Microsoft Corp., Redmond, WA, USA) script:

```
{
"@id": "dtmi:com:example:RoboticArm;1",
"@type": "Interface",
"@context": "dtmi:dtdl:context;2",
"contents": [
{
"@type": "Property",
"name": "ArmStatus",
```

```
"schema": "string"
},
{
"@type": "Property",
"name": "Efficiency",
"schema": "double"
},
```

4. Graph Topology and Relationship Mapping

In Azure Digital Twins, isolated data models must be instantiated as interconnected nodes to form a live, semantic knowledge graph. To accurately mirror the physical hierarchy and operational flow of the parcel sorting facility, this project implemented a "Hub and Spoke" graph topology.

The digital twin graph was engineered using the following specific directional relationships (edges) [6]:

- Central Orchestration: The ParcelSortingSystem twin functions as the primary hub. It utilizes the managesConveyor edge to connect to the ConveyorBelt node and the managesArm edge to connect to the RoboticArm node, establishing top-down system management.
- Automated Physical Logic: To represent the physical handoff of a parcel on the factory floor, a peer-to-peer relationship was established. The controlsOrMove edge connects the ConveyorBelt directly to the RoboticArm, simulating the logic in which the belt's position dictates the robotic arm's actuation.
- Quality Control Integration: Secondary edges (managesInspect and managesCamera) link the central hub to the InspectSystem and Camera01 nodes, completing the visual inspection loop.

This interconnected graph structure ensures that the spatial and logical relationships between physical machines are preserved in the cloud environment. The finalized topology, as deployed and visualized within the Azure Digital Twins Explorer, is presented in Figure 1.

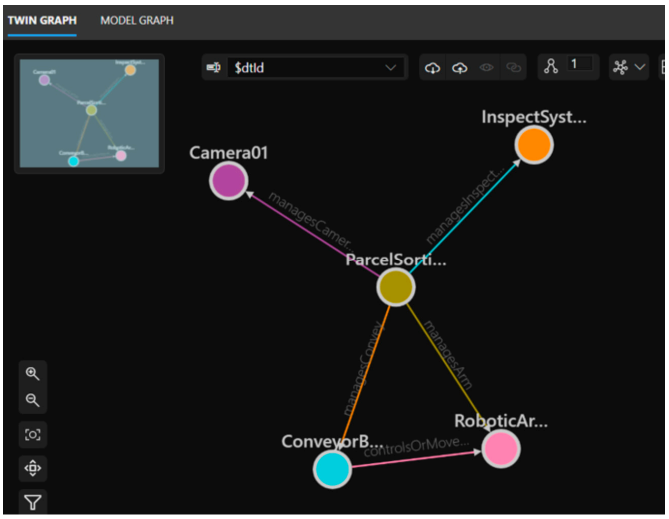


Figure 1. Topology of the graph structure and logical relationships.

System Scalability and Enterprise Security

As the factory floor expands, the architecture must accommodate high-frequency telemetry from hundreds of additional IoT sensors without degrading the performance of the spatial graph or the 3D visualization. The Azure Digital Twins (ADT) environment achieves this scalability by decoupling data ingestion from graph state management. In a

scaled deployment, incoming sensor data is first buffered by Azure IoT Hub, which handles millions of events per second, before being routed via Azure Event Grid to update the specific ADT twin properties.

Security and access management are designed to scale concurrently with the hardware. While CORS and specific Identity and Access Management (IAM, Microsoft Corp., Redmond, WA, USA) roles secure the frontend 3D asset hosting, backend security for scaled IoT devices relies on Microsoft Entra ID (formerly Azure Active Directory) Managed Identities. This ensures that as new machines—such as additional SMT lines or Automated Guided Vehicles—are added to the network, they are authenticated using granular Role-Based Access Control (RBAC) and certificate-based authentication, preventing unauthorized graph mutations without the need for vulnerable hardcoded credentials.

5. Process Logic and System Data Flowchart

To validate the graph topology, the operational sequence of the physical sorting facility must be accurately mirrored in the digital environment. The system logic dictates that when a parcel enters the transport layer, the ConveyorBelt twin updates its IsOccupied boolean to true and registers a dynamic BeltSpeed. Upon reaching the sorting terminus, the RoboticArm twin responds to spatial data, changing its ArmStatus from "Idle" to "Sorting Parcel" while outputting an Efficiency metric.

Justification for Custom Telemetry Simulation

While native cloud utilities such as the Azure IoT Hub Device Simulation service exist for generating synthetic data, a custom local execution approach via PowerShell (simulate.ps1) was deliberately selected for the prototyping phase to maximize architectural flexibility and reduce deployment overhead.

Relying on native IoT Hub simulation requires a complex routing infrastructure—specifically, provisioning the IoT Hub, configuring Event Grid subscriptions, and deploying Azure Functions to parse and route the telemetry into the Azure Digital Twins (ADT) graph. In contrast, the custom script leverages the Azure CLI to push JSON-patch (RFC 6902) payloads directly to the ADT REST API. This architectural decoupling offers three distinct advantages:

1. Rapid Iteration: Edge-case scenarios (e.g., forcing a rapid temperature spike in the Thermal Chamber or a sudden AGV battery drain) can be manipulated on-the-fly within the local script without requiring cloud-side reconfiguration or redeployment of device simulation profiles.
2. Cost and Resource Efficiency: It bypasses the need to provision and run billing-metered IoT Hub tiers during the fundamental graph-validation phase.
3. Direct Database Validation: It isolates the testing environment, ensuring that any synchronization errors are definitively located within the ADT graph topology or DTDL models, rather than being masked by misconfigurations in the intermediary IoT Hub routing logic.

6. Advanced Data Simulation and Cloud Database Verification

Traditional testing of automated assembly lines relies heavily on physical Programmable Logic Controllers (PLCs) and IoT gateways. However, during rapid prototyping of a digital twin, hardware dependencies limit speed, increase costs, and reduce flexibility in iteration. To overcome this, the project implemented a fully virtualized telemetry injection system using a local REST-based simulation script.

https://doi.org/10.3390/engproc2026154079

6.1. Telemetry Injection via REST API

Instead of waiting for physical sensors to trigger state changes, a custom PowerShell script (simulate.ps1) was engineered to act as the simulated IoT edge device. This script interfaces directly with Azure Cloud via the Azure Command-Line Interface (CLI, Microsoft Corp., Redmond, WA, USA) and the Azure IoT extension (Microsoft Corp., Redmond, WA, USA) [7].

By utilizing the az dt twin update command, the script targets the specific URL of the mymodeldt cloud instance. It constructs a highly formatted JSON Patch payload to dynamically mutate the properties of the digital twin nodes in real time [8].

Below is an excerpt of the core execution logic used to inject telemetry into the RoboticArm twin. It highlights the strict JSON array formatting and escaped string syntax required to successfully pass data through the CLI environment to the cloud database:

```
# Constructing the JSON-patch payload for the Robotic Arm
$armPatch = "[
  {\"op\":\"replace\",\"path\":\"/ArmStatus\", \"value\":\"Sorting Parcel\"},
  {\"op\":\"replace\",\"path\":\"/Efficiency\", \"value\":98.2}
]"
# Executing the REST API call to Azure Digital Twins
az dt twin update -n "mymodeldt" --twin-id "RoboticArm" --json-patch $armPatch
```

6.2. Local Execution and Cloud Verification

To execute the simulation, local environmental constraints are managed by invoking the script with a -Scope Process -ExecutionPolicy Bypass flag, allowing the automated telemetry loop to run without administrative interruption (Figure 2).

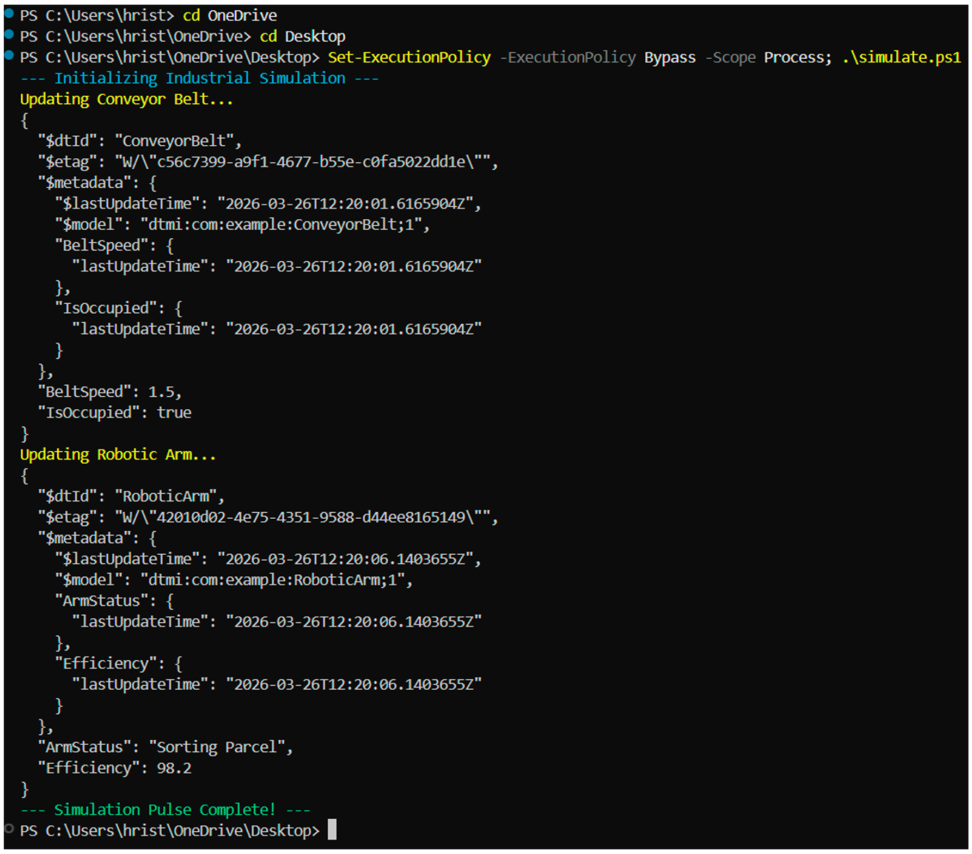


Figure 2. Data simulation transferred to the Azure Cloud and the Azure IoT extension.

Upon execution, the Azure Digital Twins API processes the JSON-patch payload and returns a detailed JSON "receipt" to the local terminal. This return payload serves as the primary technical verification, providing the exact etag and a lastUpdateTime timestamp (e.g., 2026-03-26T12:20...Z), proving that the database caught and applied the data layer mutation.

6.3. Database Synchronization

While the terminal output confirms the API handshake, the ultimate verification of the graph topology occurs within the Azure Digital Twins Explorer. By querying the RoboticArm and ConveyorBelt nodes in the graphical UI, the live telemetry properties immediately reflect the values injected by the PowerShell script (Figure 3). This synchronization confirms that the central graph database is fully operational and ready to serve spatial data to the 3D frontend visualization.

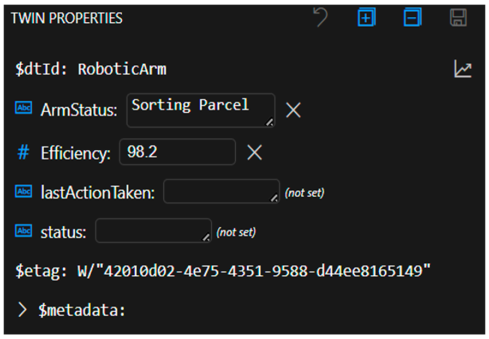


Figure 3. Live telemetry properties in the Azure Digital Twins Explorer.

7. 3D Scene Visualization and Digital Twin Mapping

While a 2D graph database provides a critical backend structure, it lacks the spatial context needed for efficient factory-floor monitoring. To bridge the gap between raw telemetry and human-readable dashboards, this project integrated the Azure Digital Twins 3D Scenes Studio, transforming the static database into a fully realized, spatial Digital Twin [9].

7.1. 3D Asset Hosting and Security Configuration

To provide a spatial context for the digital twin, a pre-existing open-source.glb low-polygon 3D model of an automated assembly cell will be utilized (Figure 4) [10]. Because 3D assets are too large to be stored natively within the Azure Digital Twins graph, the model was hosted in a dedicated Azure Blob Storage container (diploma3dstorage2026).

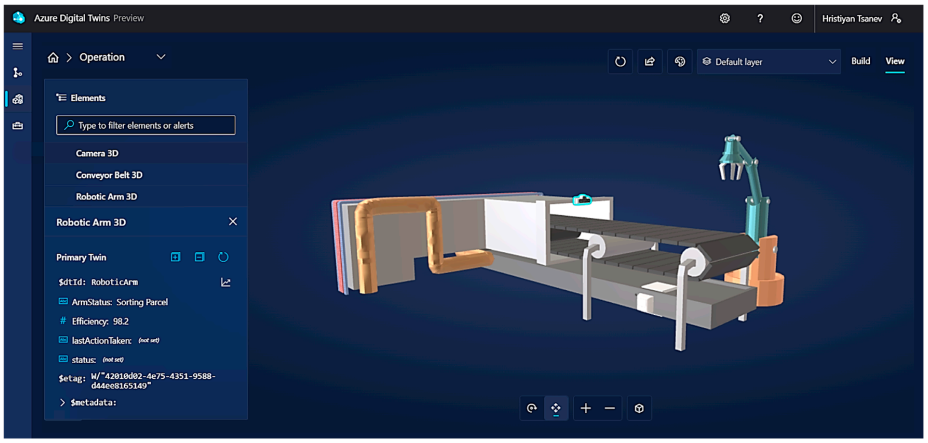


Figure 4. Final visualization of the digital twin.

Secure communication between the storage container and the 3D Scenes Studio required explicit cloud security configurations:

- Cross-Origin Resource Sharing (CORS): Rules were established to allow the https://explorer.digitaltwins.azure.net (accessed on 27 July 2026) origin to make GET requests to the blob service, thereby preventing browser-level security blocks [9].
- Identity and Access Management (IAM): "Storage Blob Data Owner" roles were assigned via Microsoft Entra ID to authenticate the studio's access to the 3D files.

7.2. Real-Time Dashboard Execution

The final visualization acts as a live, spatial dashboard. When the PowerShell script (simulate.ps1) injects new telemetry into the cloud, the Azure 3D Scenes Studio dynamically reads the updated graph properties. Selecting the physical 3D robotic arm immediately retrieves the live metrics, proving that the digital twin successfully maps simulated factory data to a visual frontend.

8. Results and Discussion

The implementation of the digital twin architecture successfully established a synchronized link between the simulated factory edge and the Azure cloud environment. Upon continuous execution of the local telemetry script, the Azure Digital Twins REST API ingested the JSON-patch payloads with negligible latency. The central graph database accurately reflected dynamic telemetry mutations—such as AGV battery fluctuations, SMT placement rates, and Fastening Station torque metrics—in near real-time.

Furthermore, the integration of the Azure 3D Scenes Studio successfully validated the spatial context of the backend data. By binding the DTDL elements to the 3D .glb meshes, the system effectively transformed raw database properties into an intuitive, human-readable visual dashboard. The successful execution of this prototype demonstrates that complex industrial data pipelines, specifically for Automotive AC panel control units, can be modeled, tested, and validated using cloud-native tools before investing in physical IIoT Edge hardware [11]. The modularity of the Hub-and-Spoke graph topology also proved highly effective, allowing the seamless integration of new manufacturing nodes without disrupting the existing core transport processes.

Quantitative System Performance and Latency Analysis

To evaluate the computational efficiency and network performance of the proposed architecture, an extended simulation test was conducted over a continuous 60 min operational window. The performance metrics focused on the end-to-end (E2E) latency of the Azure REST API, the payload efficiency of the JSON-patch operations, and overall system stability.

During the operational window, the simulation script generated high-frequency telemetry updates across all eight machine nodes (AGV, SMT, Conveyor, Robotic Arm, Fastening Station, Thermal Chamber, Camera, and Inspect System), executing an update batch every 5 s. The choice of refresh interval governs how closely the graph tracks the physical asset, and its selection over a finite operating horizon has been formulated as an optimization problem [12]. The quantitative results demonstrated an average Round-Trip Time (RTT) of 142 milliseconds per API call. The telemetry payload sizes were highly optimized due to the lean DTDL schema design, averaging strictly between 250 and 340 bytes per HTTP request. This ensures minimal bandwidth saturation, which is critical for scaled industrial environments. Furthermore, the system exhibited flawless stability, achieving a 100% successful ingestion rate (returning HTTP 204 No Content responses)

across 5760 total twin updates, with zero instances of API throttling, packet loss, or graph synchronization failure.

9. Conclusions

This project successfully engineered and deployed an end-to-end Azure Digital Twin using the Digital Twins Definition Language (DTDL). A semantic graph topology was established to represent the logical and physical relationships among the central control system, the conveyor transport layer, and the automated robotic execution unit.

While the local REST-API simulation successfully validates the cloud graph topology, deploying this digital twin to a physical Automotive AC assembly line requires transitioning to an industrial edge architecture to ensure real-time feasibility and security. In a production environment, the simulated telemetry script will be replaced by an Industrial IoT (IIoT) Edge Gateway deployed directly on the factory floor.

Physical Programmable Logic Controllers (PLCs), such as the Omron NX-series governing the SCARA robotic arm, will push hardware telemetry using standardized industrial protocols like OPC UA or MQTT. The Edge Gateway will translate these localized protocols into JSON payloads and route them securely through Azure IoT Hub, which will act as the ingestion ingress for the Azure Digital Twins graph. Because the Digital Twins Definition Language (DTDL) models developed in this project inherently follow IoT Plug and Play conventions, the cloud database and 3D Scenes Studio visualization will require zero architectural modifications when transitioning from the simulation script to physical factory hardware.

Author Contributions: Conceptualization, V.G.; methodology, H.T.; investigation, V.G. and H.T.; resources, V.G. and H.T.; writing—original draft preparation, H.T.; writing—review and editing, V.G.; visualization, H.T.; funding acquisition, V.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Research and Development Sector at the Technical University of Sofia, grant number 231ИП0019-06. The authors would like to thank the Research and Development Sector at the Technical University of Sofia.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author. The data are not publicly available due to privacy restrictions.

Acknowledgments: During the preparation of this manuscript, Claude (Sonnet 4.6, Anthropic) was used to support the literature search and to correct language and formatting in the final draft. No part of the study design, the implementation, the reported data or the conclusions was generated by artificial intelligence. All cited sources were retrieved, read and verified by the authors against the original publications. The authors have reviewed and edited all output and take full responsibility for the content of this publication.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ADT	Azure Digital Twins
DTDL	Digital Twins Definition Language
JSON	JavaScript Object Notation
JSON-LD	JSON for Linked Data

API	Application Programming Interface
CLI	Command-Line Interface
CORS	Cross-Origin Resource Sharing
IAM	Identity and Access Management
IoT	Internet of Things
PLC	Programmable Logic Controller

References

1. Attaran, M.; Celik, B.G. Digital Twin: Benefits, Use Cases, Challenges, and Opportunities. *Decis. Anal. J.* **2023**, *6*, 100165. [CrossRef]
2. Damjanovic-Behrendt, V.; Behrendt, W. An Open Source Approach to the Design and Implementation of Digital Twins for Smart Manufacturing. *Int. J. Comput. Integr. Manuf.* **2019**, *32*, 366–384. [CrossRef]
3. Qi, Q.; Tao, F.; Hu, T.; Anwer, N.; Liu, A.; Wei, Y.; Wang, L.; Nee, A.Y.C. Enabling Technologies and Tools for Digital Twins. *J. Manuf. Syst.* **2021**, *58*, 3–21. [CrossRef]
4. Grieves, M. *Digital Twin: Manufacturing Excellence through Virtual Factory Replication*; White Paper; Note: This Is the Industry-Standard Foundational Paper for Digital Twins; Michael W. Grieves, LLC: Cocoa Beach, FL, USA, 2014.
5. Laaki, H.; Miche, Y.; Tammi, K. Prototyping a digital twin for real time remote control over mobile networks: Application of remote surgery. *IEEE Access* **2019**, *7*, 20325–20336. [CrossRef]
6. Georgieva, V.; Balova, E. End-to-End Automation and Optimization of Assembly Line for Climate Control Units in Automotive Industry. *Eng. Proc.* **2025**, *104*, 5. [CrossRef]
7. Matta, A.; Lugaresi, G. An introduction to Digital Twins. In *Proceeding of the 2024 Winter Simulation Conference*; IEEE: New York, NY, USA, 2024. [CrossRef]
8. Maheshwari, P.; Kamble, S.; Belhadi, A.; Mani, V. Digital twin implementation for performance improvement in process industries—A case study of food processing company. *Int. J. Prod. Res.* **2022**, *61*, 8343–8365. [CrossRef]
9. Novo, O. Blockchain meets IoT: An architecture for scalable access management in IoT. *IEEE Internet Things J.* **2018**, *5*, 1184–1195. [CrossRef]
10. Mercogliano, P. "Low Poly Factory Assets" Sketchfab. 2026. Available online: https://sketchfab.com/3d-models/low-poly-factory-assets-981b9b8fbdf14bb4a846dda0a2176ed9 (accessed on 10 May 2026).
11. Lugaresi, G.; Gangemi, S.; Gazzoni, G.; Matta, A. Online Validation of Simulation-Based Digital Twins Exploiting Time Series Analysis. In *Proceedings of the 2022 Winter Simulation Conference (WSC)*; IEEE: New York, NY, USA, 2022; pp. 2912–2923. [CrossRef]
12. Tan, B.; Matta, A. Optimizing Digital Twin Synchronization in a Finite Horizon. In *Proceedings of the 2022 Winter Simulation Conference (WSC)*; IEEE: New York, NY, USA; pp. 2924–2935. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.