2025 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE
MUNICH, GERMANY
OCTOBER 14-15, 2025

Harnessing a Digital Twin for Personalized Type-1 Diabetes Care: A Model for Glucose Control and Insulin Administration

Ammar Abdelghany[1], Mohamed Hamed[2], Asmaa Khaled[3], Nada M. Alfowey[4], Tamer Basha[5], Loai Ali[6], Mohamed AbdElSalam[7]
[1,2,3,4]Biomedical Engineering Department, Cairo University, Cairo, Egypt ({ammar.abdelghany74, mohamed.a.hamed00109, asmaahaamza, nada.alfowey2}@gmail.com)
[5]Biomedical Engineering Department, Cairo University, Cairo, Egypt
tamer.basha@eng1-cu.edu.eg
[6,7]Siemens Digital Industries Software, Cairo, Egypt ({loai.ali, mohamed.abd_el_salam_ahmed}@siemens.com)

Abstract— **Managing Type 1 Diabetes (T1D) is a complex and demanding process, requiring continuous glucose monitoring and precise insulin administration to maintain optimal glycemic control. In this paper, we introduce an innovative approach to diabetes management by leveraging digital twin technology to develop and verify a closed-loop artificial pancreas system. The approach used a compositional simulation interconnect framework to seamlessly integrate a Runtime Verification (RV) monitor for property checking, advanced control algorithms, and hardware-in-the-loop (HIL) testing. By automating insulin delivery and ensuring real-time safety verification, this system aims to enhance glycemic control, reduce patient burden, and improve overall quality of life for individuals with T1D. The integration of digital twin technology in the medical vertical represents a significant advancement in the development of reliable and patient-centric diabetes management systems.**

Keywords*— *Digital Twin simulation, Type 1 Diabetes, Closed-loop control, Runtime verification.

I. Introduction

Digital Twin (DT) technology [1], originally rooted in manufacturing, has emerged as a valuable tool for simulating and controlling complex systems in real time. By replicating physical systems virtually, DTs enable predictive modeling, closed-loop experimentation, and risk-free validation—capabilities especially relevant in healthcare. Recent advances have positioned DTs as enablers of personalized medicine, allowing treatment strategies to be tested and tuned per patient before deployment.

Type 1 Diabetes Mellitus (T1D) is a prime application, being a chronic autoimmune condition where insulin production is absent [2]. Despite advances in continuous glucose monitors (CGMs) [3] and insulin pumps, achieving stable glycemic control remains difficult due to individual variability and lifestyle factors like meals, exercise, and stress [4].

We propose a DT-based framework for simulating a personalized artificial pancreas system. It integrates (i) a virtual patient model exported as a Functional Mock-up Unit (FMU) [5], (ii) an adaptive insulin controller responsive to meals and physical activity, and (iii) a runtime safety monitor using TP-DejaVu [6].

These components are orchestrated via a compositional interconnect fabric, enabling pre-clinical validation, real-time safety verification, and closed-loop control—all within a unified simulation platform designed for reliability and patient-centric tuning.

II. Background: Standards & Tools for Co-simulation

This section presents the key standards and tools that form the foundation of our co-simulation architecture for the diabetes management digital twin. Effective co-simulation relies on robust interoperability mechanisms and specialized platforms to integrate heterogeneous system components [7].

Functional Mock-up Interface (FMI)

A widely adopted standard for enabling co-simulation and model exchange is the FMI [8]. FMI defines a standardized Application Programming Interface (API) and a packaging format for model components, known as Functional Mock-up Units (FMUs). An FMU encapsulates a model (our virtual patient model developed in Simcenter Amesim) as a black box that exposes its functionality through the methods defined in the FMI API.

Compositional Simulation Interconnect Framework

We leverage an optimized interconnect framework [9] that enables the integration of heterogeneous components to build domain-specific digital twins across verticals such as automotive, robotics, avionics, and healthcare.

This framework supports the co-simulation of diverse clients, including sensor/scenario simulators, mechatronic system models, cloud services, compute nodes, and software modules implemented at various abstraction levels (e.g., C/C++, RTL, SystemC). It can also interface with physical hardware (e.g., embedded boards) over standard communication links such as Ethernet and CAN. The design methodology is centered on three core enablers: the Gateway mechanism, the Digital Twin Description Language (DTDL), and an automation flow that simplifies system integration, as illustrated in Figure 1.

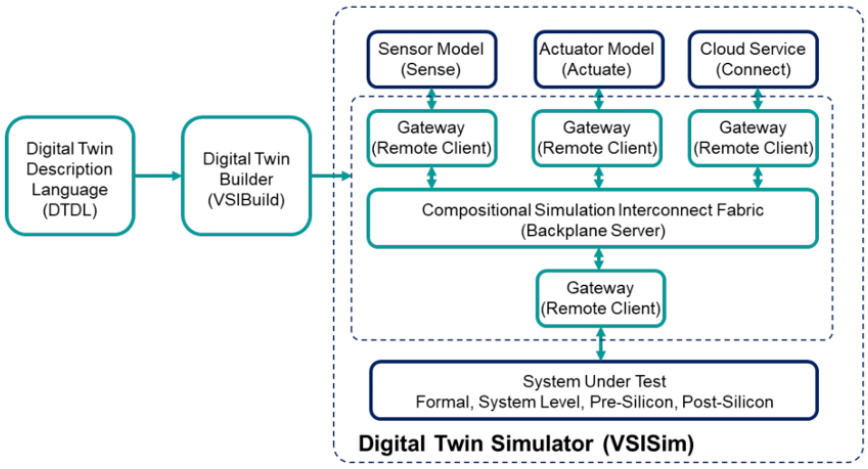
Digital Twin Description Language (DTDL)
Digital Twin Builder (VSIBuild)
Sensor Model (Sense)
Actuator Model (Actuate)
Cloud Service (Connect)
Gateway (Remote Client)
Gateway (Remote Client)
Gateway (Remote Client)
Compositional Simulation Interconnect Fabric (Backplane Server)
Gateway (Remote Client)
System Under Test
Formal, System Level, Pre-Silicon, Post-Silicon
Digital Twin Simulator (VSISim)

Fig. 1: Digital Twin Interconnect Framework.

A. Gateway Concept

Each digital twin component is connected to the VSI backplane server via a dedicated *Gateway*, which handles data conversion, routing, and protocol interfacing. Gateways abstract away communication complexity by translating between simulated or physical transport protocols and the internal interface of the component. VSI categorizes gateways into several types, including Language2Protocol (e.g., C++, Python, ROS), Pre-Silicon (VPs, RTL), Post-Silicon (HiL setups), and Specialized (sensor/actuator interfaces, cloud platforms, or external tools). Supported communication protocols include Ethernet, CAN, LIN, and AXI, among others.

B. Digital Twin Description Language (DTDL)

DTDL is a domain-specific intermediate format used to define the connectivity and interaction between components in the digital twin system. It acts as the blueprint for system composition and enables automatic generation of the required infrastructure. Due to space limitations, we omit the detailed syntax and semantics of DTDL in this paper.

C. Automation Flow

To simplify the setup of the co-simulation environment, VSI provides an automation pipeline for generating the backplane server and associated gateways. The user begins by describing the system architecture in DTDL. This description is parsed by a builder tool that generates the interconnect configuration. Once the System Under Test (SUT) is plugged in, the simulation can be launched via the VSI simulator.

The automation flow includes the following tools:

- **VSIBuild:** A command-line utility that parses DTDL files and generates the skeleton configuration for a digital twin. This greatly reduces manual integration overhead and enables rapid prototyping. Users invoke it with: `vsiBuild [options] -f <commandFile>`.
- **VSISim:** A simulator that manages time synchronization across all components and provides an interface for interactive control. It spawns terminal windows for each connected client and provides simulation control via a command prompt. The simulation is launched using: `vsiSim <digital_twin_name.dt> [options]`.

III. Related Work

Several models simulate glucose-insulin dynamics in Type 1 Diabetes (T1D), from the Bergman Minimal Model [10] to more detailed representations by Sorensen [11] and Cobelli [12]. The UVA/Padova simulator [13] and Hovorka model [14] strike a balance between detail and usability. We use the Hovorka model to construct a virtual patient in Simcenter Amesim, exported as an FMI-compliant Functional Mock-up Unit (FMU), enabling integration into a closed-loop digital twin co-simulation.

To handle variability and insulin delay, we implement a neural network–based PID (NN-PID) controller [15], with an LSTM predicting PID gains (K_p, K_i, K_d) from real-time glucose input. The patient model and NN-PID controller operate within a co-simulation framework that synchronizes time and data exchange.

To enhance safety, we integrate the TP-DejaVu runtime monitor to verify temporal properties during simulation. A test case manager introduces clinically relevant edge cases, supporting traceability, early validation, and functional safety for embedded insulin control systems.

IV. Architecture and Workflow

This section presents the overall architecture of our digital twin system designed for closed-loop management of Type-1 Diabetes (T1D). The proposed system integrates four principal components, each playing a distinct role in the real-time simulation, control, and verification pipeline. These components include a virtual patient model implemented in Simcenter Amesim, a neural network-tuned PID controller for insulin delivery, a runtime verification engine based on TP-DejaVu, all connected using a compositional simulation interconnect framework.

A. System Overview

The complete digital twin (DT) system functions as a hardware-software co-simulation platform capable of continuous closed-loop operation. Each subsystem acts as a remote client connected to the interconnect backplane. The simulation flow and data exchange are managed by a DT configuration file written in DTDL, which the VSI builder then parses. This setup ensures time synchronization and signal mapping for inter-process communication.

Figure 2 illustrates the system-level architecture. The virtual patient receives insulin infusion rates as I/O glucose levels every simulation step. The glucose signal is passed to the NN-PID controller, which computes the appropriate insulin rate. Both signals are recorded and analyzed in real-time by the TP-DejaVu, which verifies temporal safety and stability properties. Communication across the components is managed via TCP-based protocol through the interconnect backplane.

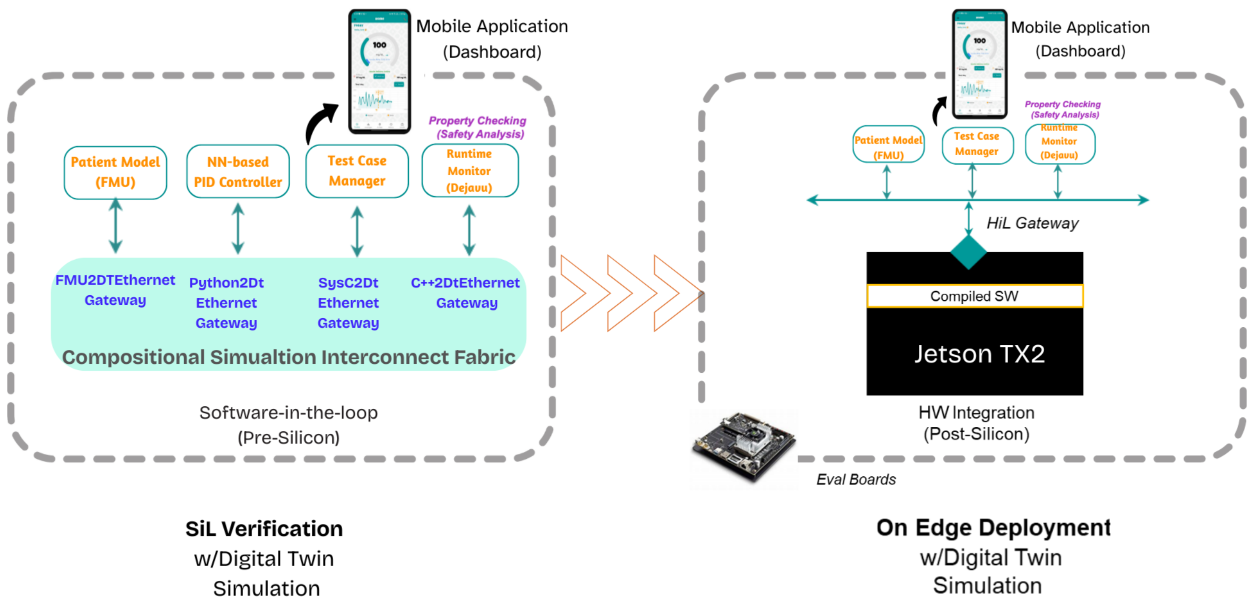
Mobile Application (Dashboard)
Property Checking (Safety Analysis)
Patient Model (FMU)
NN-based PID Controller
Test Case Manager
Runtime Monitor (Dejavu)
FMU2DTEthernet Gateway
Python2Dt Ethernet Gateway
SysC2Dt Ethernet Gateway
C++2DtEthernet Gateway
Compositional Simualtion Interconnect Fabric
Software-in-the-loop (Pre-Silicon)
SiL Verification w/Digital Twin Simulation
Mobile Application (Dashboard)
Property Checking (Safety Analysis)
Patient Model (FMU)
Test Case Manager
Runtime Monitor (Dejavu)
HiL Gateway
Compiled SW
Jetson TX2
HW Integration (Post-Silicon)
Eval Boards
On Edge Deployment w/Digital Twin Simulation

Fig. 2: Architecture of the closed-loop T1D digital twin using VSI interconnect

As shown in Figure 3, we defined two communication ports using the DTDL, assigning suitable gateway types to each. This configuration involved specifying the signal names, their data directions (input/output), and establishing the data exchange mechanism based on the selected communication protocol—in our setup, the physical Ethernet protocol.

```
# Add the FMU component

add component -type Fmu -name patientModel -fmuPath ‹ Path to FMU >
add port -name patientModelPort -gateway fmu2DtEthernet -componentName patientModel

# Add the TP dejavu component

add component -type C++ -name tpDejavu

add port -name tpDejavuPort -gateway c++2DtEthernet -componentName tpDejavu

# Add the Controller component

add component -name controller -type Python -remoteComponent
add port -name controller -gateway python2DtEthernet -componentName controller
```

Fig. 3: Digital Twin Description Language for digital twin configuration.

B. Virtual Patient Model (FMU)

At the core of our digital twin lies a high-fidelity virtual patient model that replicates glucose-insulin dynamics in individuals with T1D. This model is implemented using Simcenter Amesim and builds upon the widely used Hovorka model, which simulates insulin pharmacokinetics (PK) and glucose pharmacodynamics (PD). Our implementation extends this foundation by incorporating modules for physical activity (exercise) and circadian variation in insulin sensitivity to reflect real-world patient behavior more accurately.

The virtual patient is exported as an FMU compliant with the FMI 2.0 co-simulation standard, enabling time-synchronized interaction with external components such as the insulin controller, test case generator, and runtime monitor. The FMU operates on a time-driven simulation basis with a fixed step size of 1 minute, during which it processes control signals (insulin dosing, meal intake, and activity flags) and updates physiological states accordingly.

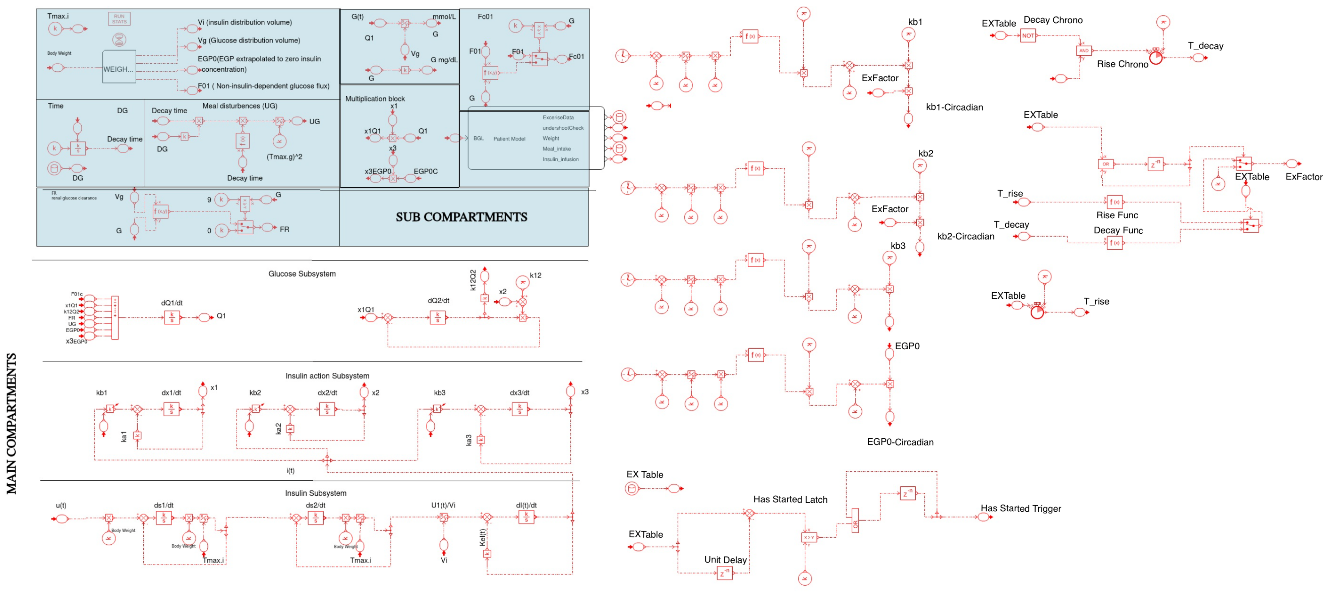

Fig. 4: Virtual Patient Model in Amesim.

Internally, the model comprises multiple interacting subsystems that simulate insulin absorption, glucose appearance from meals, endogenous glucose production, tissue glucose uptake, and renal clearance. These are governed by a set of nonlinear differential equations parameterized by patient-specific characteristics (body weight, insulin sensitivity, CHO-to-insulin ratio). The exercise module triggers transient increases in glucose disposal rate and suppresses hepatic glucose production, simulating the physiological response to moderate aerobic activity.

Table I outlines the key input and output signals of the FMU. These are exchanged at runtime with other digital twin components through VSI to facilitate closed-loop glucose control.

TABLE I: Inputs and Outputs of the Virtual Patient Model (FMU)

Signal	Description
Inputs	
Insulin Infusion Rate (U/min)	Continuous input representing either basal or bolus insulin delivered under the skin. Used to update insulin compartments.
CHO (Carbohydrate) Intake (g)	Discrete carbohydrate intake events representing meals.
Exercise Start/Stop Flag	Boolean signal triggering increased insulin sensitivity and modified glucose dynamics to simulate physical activity.
Outputs	
Sensor Glucose (mg/dL)	Simulated CGM signal, derived from plasma glucose concentration. Used by the controller as the primary feedback signal.

C. Neural Network-Tuned PID Insulin Controller

Maintaining normoglycemia in T1D requires timely insulin dosing in response to dynamic glucose fluctuations. We employ a classical PID structure augmented with adaptive neural network tuning.

Control Objectives. The controller regulates insulin via:

- **Basal insulin:** Continuous delivery for background metabolic needs.
- **Bolus insulin:** Discrete doses to mitigate meal- or excursion-induced spikes.

PID Control Law. The control signal $u(t)$ is given by:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de(t)}{dt} + u_{\text{corr}}(t) \tag{1}$$

where $e(t)$ is the glucose error (setpoint minus CGM value),
and $u_{\text{corr}}(t)$ is a correction term applied during bolus events.

Adaptive Gain Tuning via LSTM. To account for intra- and inter-patient variability, a Long Short-Term Memory (LSTM) network dynamically predicts optimal PID gains every 5 minutes, based on recent CGM history and time-of-day features. The architecture includes two LSTM layers (64 and 32 units) and a dense output layer trained on the OhioT1DM dataset [16]. Gains are optimized for postprandial stability and glycemic range compliance.

Reinforcement Learning Alternative. We also explore a reinforcement learning (RL) strategy using an Advantage Actor-Critic (A2C) agent that continuously adjusts (K_p, K_i, K_d) in a continuous action space. The agent observes glucose levels, rates of change, and time-of-day; rewards prioritize time-in-range (70–180 mg/dL), penalizing hypoglycemia and severe excursions. The policy network consists of actor-critic pairs (64/32 units each), trained via policy gradient. Safety constraints (insulin lockout, episode termination for <50 mg/dL) ensure robust behavior.

Deployment & Safety. The LSTM controller has 29k trainable parameters, 115 KB memory footprint, and $<$ 10 ms inference latency, suitable for embedded platforms like **NVIDIA Jetson TX2**. Basal insulin is continuously updated; bolus insulin is delivered through event-based triggers. Safety caps on insulin dosing minimize hypoglycemia and ensure real-time reliability when interfaced with the virtual patient (Section IV-B).

D. Mobile Application Dashboard

To visualize the digital twin's output data in real time, we developed a lightweight mobile application that functions as a dashboard. The app receives structured simulation data (e.g., blood glucose levels, insulin doses, and detected events) via a backend interface and displays them using interactive charts and time series plots.

The dashboard, as illustrated in figure 5, provides intuitive views for tracking key metrics such as time-in-range, bolus history, and glucose trends. It was built using Flutter for cross-platform compatibility and communicates with the simulation environment through a lightweight API. This enables quick feedback during co-simulation and allows end-users or clinicians to monitor outcomes on-the-go.

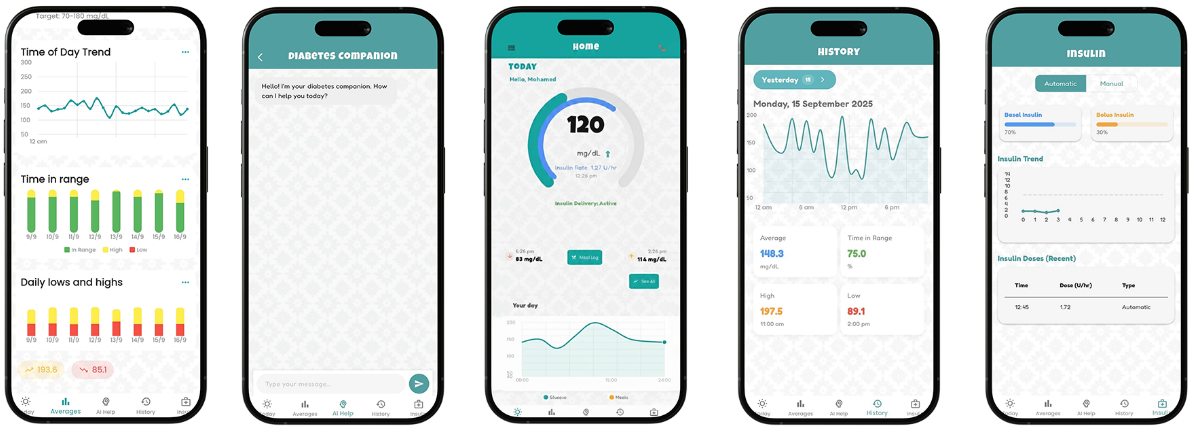

Fig. 5: Mobile application screens

E. *Runtime Monitoring using TP-DejaVu*

To ensure real-time safety in insulin delivery, we employ TP-DejaVu, an advanced runtime verification (RV) framework for temporal property checking. It monitors simulation traces during closed-loop operation and flags violations of user-defined safety constraints. The tool supports property specification in two layers:

- **Operational Specification:** Defines intermediate variables and computations from incoming event traces.
- **Declarative Specification:** States properties using temporal logic over those variables (bounded response, past-time logic).

F. *Safety Property Definitions*

1) **Hypoglycemia Hazard:** Ensures blood glucose remains above 50 mg/dL.

```
prop MonitorInsulin: exists glucoseLevel . (glucoseLevel < 50)
```

2) **Hyperglycemia Hazard:** Detects readings exceeding a defined upper threshold (250 mg/dL).

```
prop MonitorInsulin: exists glucoseLevel . (glucoseLevel > 250)
```

3) **Inaccurate Glucose Trends:** Flags large variations (>50 mg/dL) between two consecutive readings.

```
prop MonitorInsulin: exists deltaGlucose . (deltaGlucose > 100)
```

4) **Sensor Error Detection:** Detects biologically invalid glucose values, particularly zeros.

```
prop MonitorInsulin: exists glucoseLevel . (glucoseLevel = 0)
```

Scenario Generation

To validate the runtime monitor, we simulate a set of controlled fault injection scenarios involving patient parameters, meal times, and exercise events. Table II summarizes the scenarios used to test safety property violations and the corresponding system responses.

TABLE II: Scenario Generation

Scenario	Safety Requirement Monitored	Expected Property to be Violated	Actions Taken
0	Normal conditions with reasonable glucose levels, scheduled meals, and moderate exercise	None (Baseline)	No action required; monitor confirms system stability
I	Blood glucose level falls below 50 mg/dL (hypoglycemia danger)	Hypoglycemia hazard	Pause insulin infusion
II	Blood glucose level exceeds hyperglycemia threshold	Hyperglycemia hazard	Raise a flag
III	Variation between two consecutive readings > 50 mg/dL due to interference (e.g., microwaves)	Risk of inaccurate glucose trend	Flag as noise, smooth signal, and avoid acting on it
IV	Glucose reading is zero due to a sensor error	Sensor error	Suspend loop, trigger alert, request sensor check

V. Experimental Results

We evaluated the system in two closed-loop control configurations over a 24-hour scenario featuring three meals and one exercise event:

- **LSTM-PID:** A neural network-tuned PID controller.
- **RL-PID:** A reinforcement learning-based PID controller.

A. Glucose Dynamics

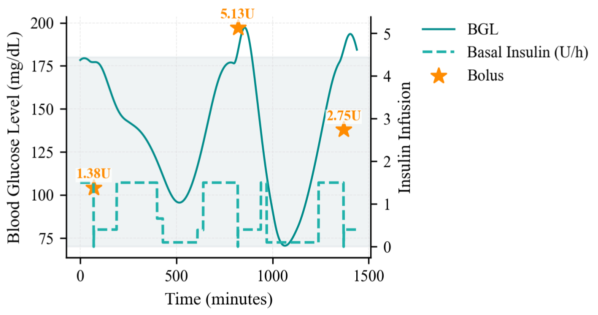

Fig. 6: Glucose response using LSTM-PID controller.

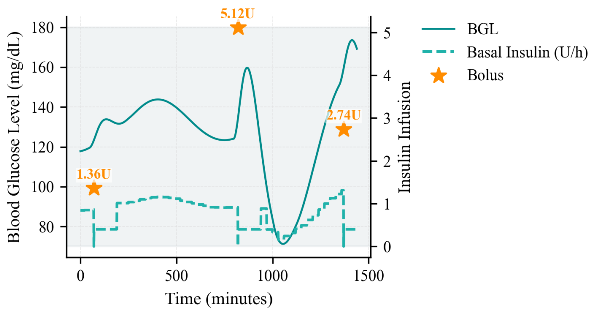

Fig. 7: Glucose response using RL-PID controller.

Both controllers successfully maintained blood glucose within the target range, with smooth recovery from disturbances and no hypoglycemia events.

B. Controller Performance Metrics

TABLE III: Performance Metrics for LSTM and RL Controllers

Metric	**LSTM-PID**	**RL-PID**
Time in Range (70–180 mg/dL)	88.82%	**100.00%**
Time Below 70 mg/dL	0.00%	0.00%
Time Above 180 mg/dL	11.18%	**0.00%**
Total Insulin Used (U)	28.08	**27.70**
Bolus Events	3	3
Hypoglycemic Events (Blood glucose level less than 70 mg/dL)	0	0

Both controllers demonstrated strong performance in managing blood glucose levels while ensuring patient safety. A key shared success metric is the complete absence of hypoglycemic events, a critical safety goal.

The RL-PID controller excels at maintaining a perfect score within the target range, achieving 100.00% Time in Range. This indicates its exceptional ability to prevent blood glucose excursions, both high and low, keeping it consistently within the optimal range. It accomplished this with a slightly lower total insulin usage of 27.70 U.

The LSTM-PID controller also showed very effective glucose management, with a high 88.82% Time in Range. It successfully maintained blood glucose within the target range for the vast majority of the time and, like its counterpart, had zero hypoglycemic events. This performance demonstrates its robust capabilities in glucose regulation and its ability to recover from disturbances while minimizing the risk of dangerously low blood sugar.

C. System Robustness and Noise Injection Analysis

Beyond evaluating the primary control performance, we stressed the system with a series of fault injection scenarios to verify its robustness and safety. This included testing against a scenario of inaccurate glucose readings caused by external interference, such as microwaves.

Figure 8, which corresponds to Scenario III in Table II, illustrates the system's behavior during a simulated noise event. During the simulation, multiple noise impulses were injected into the simulated glucose readings, causing sharp, non-physiological spikes in the Blood Glucose Level (BGL). The runtime monitor, detected these rapid, large variations (greater than 50 mg/dL) and flagged them as noise.

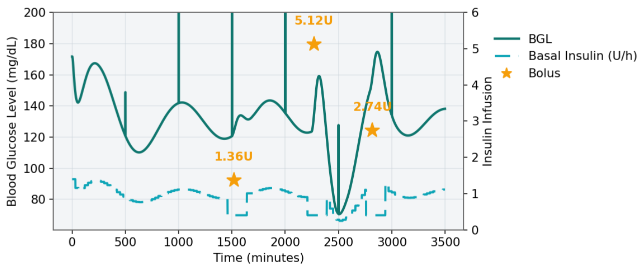

Fig. 8: Glucose response under noise injection using RL-PID controller.

This action prevented the insulin controller from overreacting and administering an inappropriate insulin basal or an excessive insulin bolus dose, which could have led to hypoglycemia. By correctly identifying and ignoring the erroneous data, the system maintained a stable output and demonstrated its ability to operate reliably in the presence of sensor disturbances. This confirms that the framework can ensure patient safety and prevent incorrect insulin dosing under real-world, non-ideal conditions. This also serves as a sign-off for the system's ability to maintain blood glucose levels within the correct range while avoiding hypoglycemic and hyperglycemic events.

D. Deployment and Hardware-in-the-Loop (HIL) Verification

After validating the system in a software-only simulation environment, we ported the control logic to the embedded NVIDIA Jetson TX2 board for HIL testing. The results were consistent, confirming that the model's performance remained stable across both software and embedded hardware platforms.

VI. Conclusion and future work

This work introduced a digital twin framework for personalized T1D management, integrating the Hovorka patient model (via FMU), an adaptive NN-PID controller, a runtime verification monitor, and scenario-based testing. The system enables real-time, safety-aware glucose regulation and supports HIL deployment. Experimental results demonstrated that the NN-PID controller adapts to dynamic inputs and maintains blood glucose levels effectively, even when deployed on embedded hardware. The proposed system offers a pathway for developing and validating safe and personalized artificial pancreas solutions.

There are several opportunities to extend this work. One direction lies in leveraging formal verification techniques with richer property-specification languages and automated proof engines, which could further strengthen the runtime verification layer and enable rigorous safety guarantees across a wider range of operating conditions. Another opportunity is the expansion of the HIL environment through the integration of additional sensor and actuator models. Such enhancements would allow more representative system-level testing and stress evaluation, improving confidence in the robustness of the closed-loop framework under realistic deployment scenarios.

References

[1] M. Grieves and J. Vickers, "Origins of the digital twin concept," *Florida Institute of Technology*, vol. 8, pp. 3–20, 2016.
[2] M. A. Atkinson, G. S. Eisenbarth, and A. W. Michels, "Type 1 diabetes," *The Lancet*, vol. 383, no. 9911, pp. 69–82, 2014.
[3] L. Heinemann and S. Freckmann, "Cgm versus fgm; or, continuous glucose monitoring is not flash glucose monitoring," *Journal of Diabetes Science and Technology*, vol. 9, no. 5, pp. 947–950, 2015.
[4] The DCCT Research Group, "The effect of intensive treatment of diabetes on the development and progression of long-term complications in insulin-dependent diabetes mellitus," *New England Journal of Medicine*, vol. 329, no. 14, pp. 977–986, 1993.
[5] Siemens Digital Industries Software, "Simcenter amesim: System simulation software," https://plm.sw.siemens.com/en-US/simcenter/systems-simulation/amesim/, 2024, accessed: 2025-07-02.
[6] DejaVu Runtime Monitor, "Dejavu runtime monitor," https://github.com/moraneus/TP-DejaVu, [Online; accessed 18-June-2025].
[7] C. Gomes, C. Thule, D. Broman, P. G. Larsen, and H. Vangheluwe, "Co-simulation: State of the art," *arXiv preprint arXiv:1702.00686*, 2017.
[8] T. Blochwitz *et al.*, "Functional mockup interface 2.0: The standard for tool independent exchange of simulation models," in *Proceedings of the 9th International Modelica Conference*, 2012, pp. 173–184.
[9] Siemens EDA, "Innexis virtual system interconnect (vsi)," https://eda.sw.siemens.com/en-US/ic/hav/innexis/virtual-system-interconnect/, [Online; accessed 18-June-2025].
[10] R. N. Bergman, Y. Z. Ider, C. R. Bowden, and C. Cobelli, "Quantitative estimation of insulin sensitivity," *American Journal of Physiology*, vol. 236, no. 6, pp. E667–E677, 1979.
[11] J. T. Sorensen, "A physiologic model of glucose metabolism in man and its use to design and assess improved insulin therapies for diabetes," Ph.D. dissertation, Massachusetts Institute of Technology, 1985.
[12] C. Cobelli and A. Mari, "A physiological model for glucose and insulin control in man," *Mathematical Biosciences*, vol. 58, no. 1, pp. 27–60, 1982.
[13] C. D. Man *et al.*, "The uva/padova type 1 diabetes simulator: New features," *Journal of Diabetes Science and Technology*, vol. 8, no. 1, pp. 26–34, 2014.
[14] R. Hovorka, F. Shojaee-Moradie, P. V. Carroll *et al.*, "Partitioning glucose distribution/transport, disposal, and endogenous production during ivgtt," *American Journal of Physiology-Endocrinology and Metabolism*, vol. 282, no. 5, pp. E992–E1007, 2002.
[15] F. Rivas-Echeverría, A. Ríos-Bolívar, and J. Casales-Echeverría, "Neural network-based auto-tuning for pid controllers," *Neural Network World*, vol. 11, pp. 2–4, 2001.
[16] C. Marling and R. Bunescu, "The ohiot1dm dataset for blood glucose level prediction: Update 2020," in *CEUR Workshop Proceedings*, vol. 2675, Sep. 2020, pp. 71–74.