2026 IEEE International Symposium on Measurement in Networking and Communications (MNC) | 979-8-3315-6214-4/26/$31.00 ©2026 IEEE | DOI: 10.1109/MNC69143.2026.11680652

IEEE 1451.0 and 1451.1.6-based Smart Transducer Digital Twin Testbed for IoT Real-Time Monitoring and Control of Smart Homes

Eugene Song
Communications Technology Laboratory
National Institute of Standards and Technology (NIST)
Gaithersburg, USA
eugene.song@nist.gov

Tyler Wong
Department of Electrical and Computer Engineering
University of Maryland, College Park,
Maryland, USA
trw4@umd.edu

Thomas Roth
Communications Technology Laboratory
National Institute of Standards and Technology (NIST)
Gaithersburg, USA
thomas.roth@nist.gov

***Abstract*— Digital twins (DTs) are virtual replicas of physical assets, systems, and processes that enable real-time monitoring and control and can be used to improve decision-making, increase efficiency, test compatibility and interoperability, and achieve cost savings across various Internet of Things (IoT) applications. This paper introduces an Institute of Electrical and Electronics Engineers (IEEE) 1451.0 and 1451.1.6-based smart transducer DT testbed for IoT real-time monitoring and control of smart homes. This testbed consists of an IEEE 1451.0 and 1451.1.6-based smart transducer physical twin (PT), one smart transducer DT, one MQTT broker, and a bidirectional information exchange between the PT and DT for real-time monitoring and control. The PT is developed using a Raspberry Pi and a sensor and actuator development kit. The DT is developed using Python programming language. The DT can communicate with the PT via the MQTT broker using IEEE 1451.0 messages and 1451.1.6 topics for real-time monitoring and controlling. A simple real-time temperature monitoring and control test case using a smart temperature sensor is used to test and validate that both PT and DT work correctly based on the IEEE 1451.0 and 1451.1.6 standards. This DT testbed will be used to test interoperability of IEEE 1451.0 and 1451.1.6-based smart transducers (sensors and/or actuators).**

Keywords—Digital Twin, IEEE 1451.0, IEEE 1451.1.6, Monitoring and Control, Physical Twin, Real-Time, Smart Actuator, Smart Home, Smart Sensor, Smart Transducer, Testbed

I. INTRODUCTION

A physical twin (PT) is a tangible physical device, process, or system with a corresponding digital twin (DT) that models or simulates its behavior. A DT is a set of virtual information constructs that mimics the structure, context, and behavior of a PT, is dynamically updated with data from its PT, and has a predictive capability that informs decisions [1]. A DT is a formal digital representation of PT, which captures attributes and behaviors of PT suitable for communication, storage, interpretation, or processing within a specific context [2]. DT technology offers numerous benefits, including increased productivity, reduced product development time and costs, improved decision-making and risk mitigation, improved product quality and development processes, and increased efficiency in various industries [3]. DT has been widely used in manufacturing, construction, healthcare, energy, automotive for improved visualization, maintenance, prediction, and interoperability [4].

Transducers (sensors and/or actuators) are widely used in various Internet of Things (IoT) applications and play a key role in real-time monitoring and control applications. The main challenges of IoT sensors and actuators include diverse connectivity, interoperability, and cybersecurity. The solutions to overcome these challenges include adopting standardized interfaces and communication protocols, conducting testing, and certification. The Institute of Electrical and Electronics Engineers (IEEE) 1451.0-2024 [5] defines common functions of IoT sensor network devices, network services, transducer services, and transducer electronic data sheet (TEDS) formats to help achieve interoperability in network and transducer interfaces. This standard also defines a universal unique identification (UUID), security framework, and time synchronization framework. IoT sensor network devices equipped with UUID, TEDS, and standardized interfaces and communication protocols can thus achieve global identity, time synchronizing, secured network communications, and interoperability. IEEE 1451.0 facilitates IoT sensor network installation, maintenance, and upgrades and reduces the total life-cycle costs of IoT sensor networks. IEEE 1451.1.6 [6] defines a standardized method for transporting IEEE 1451.0 messages over a sensor network using message queue telemetry transport (MQTT) to establish a light-weight, simplified protocol to handle IEEE 1451 network communications.

Smart transducer DT works toward common data formats and streamlined data in IoT systems, and one of most common applications of DTs is used to test and assess interoperability [4]. A DT testbed is a collaborative testing environment, where multiple stakeholders can integrate, test, and validate their DT technologies in a vendor-neutral domain. This paper mainly focuses on developing an IEEE 1451.0 and 1451.1.6-based smart transducer DT testbed for IoT real-time monitoring and control to test interoperability of IEEE 1451-based smart transducers. Related works are described in Section II. Section III introduces the IEEE 1451.0 and 1451.1.6-based smart transducer DT. Section IV describes an IEEE 1451.0 and 1451.1.6-based smart transducer DT testbed. Section V provides the testing result of the DT testbed for smart homes. The summary and conclusion are described in Section VI.

II. Related Work

An IEEE P1451.1-based smart sensor digital twin federation was introduced [7], which consists of three federates: DT Federate, DT Tester Federate, and Federation Experiment Manager. The DT Federate emulates both desired, non-linear behaviors and failure modes of an IEEE P1451.1-based smart sensor for future smart sensor federation research. Rocha et al [8], introduced an interoperable IEEE 1451 semantic DT using a JavaScript Object Notation for Linked Data context inside the network capable application processor (NCAP) to encode the data semantically by referencing an external IEEE 1451 ontology for semantic communications and interoperability. Further, an IEEE 1451-based smart transducer digital twin framework (DTF) was introduced, which consists of an IEEE 1451-based smart transducer PT using the specific embedded device, sensor, and actuator development kit, its corresponding DT, and a bidirectional information exchange between them for real-time monitoring and control. The IEEE 1451.0 and 1451.1.6-based smart transducer DT is developed using the Node-RED and message queue telemetry transport (MQTT) protocol [9].

ISO 23247 outlines a four-layer functional framework for DTs: observable elements, communication, DT, and users. This framework includes a set of protocols for creating new DTs and maintaining existed DTs [10]. ISO/IEC JTC 1, SC 41: IoT and DT Guidelines [11], which focus on IoT interoperability and standardization, provide some guidance for integrating IoT devices with digital twin platforms, which include DT maturity model, and assessment guidance, and DT use cases. The Digital Twin Consortium (DTC) describes a DT system interoperability framework that is a structured approach to enabling seamless interoperability across complex, distributed DT systems; its main goal is to drive awareness, adoption, interoperability, and development of DT technology [12]. DTC DT Testbed program [13] is a member-driven program advancing the next generation of DT systems and technologies. This testbed program has a total of 16 members that can model, simulate, integrate, test, verify, deploy, and optimize digital twin solutions. For example, Twinsense testbed is a virtual sensing platform to enhance real-time understanding and improve learning systems, which demonstrates how DT technology can provide real-time virtual measurements of critical variables across industrial assets [14].

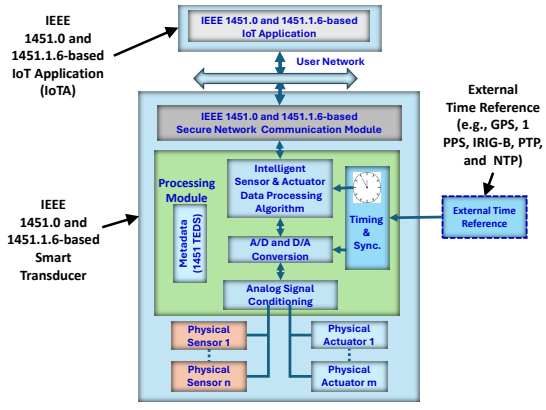


Fig. 1. IEEE 1451.0 and 1451.1.6-based Smart Transducer.

These previously mentioned DTs based on the IEEE, ISO, and ISO/IEC standards, and Twinsense testbed [14] do not focus on testing interoperability, instead of DT modeling, simulation, verification, deployment, and optimization. Based on the DT framework [9], this paper mainly focuses on new design and development of an IEEE 1451.0 and 1451.1.6 standards-based smart transducer DT testbed using popular Raspberry Pi and sensor kit and Python for IoT real-time monitoring and control applications. This includes fleshed out command and reply messages structures, credential validation logic that authenticates messages prior to processing, as well as a Python-based graphic user interface (GUI) that records and displays message data. This testbed will be used to test interoperability of IEEE 1451.0 and 1451.1.6 smart sensors and actuators provided by diverse manufacturers and vendors in the future.

III. IEEE 1451.0 and 1451.1.6-based Smart Transducer Digital Twin

A. IEEE 1451.0 and 1451.1.6-Based Smart Transducer

Fig. 1 shows an IEEE 1451.0 and 1451.1.6-based smart transducer that consists of a set of physical transducers (sensors and/or actuators), a processing module, TEDS, and secure network communication module. Including sensing and actuating capabilities, the smart transducer has the following more capabilities [15]:

- signal processing,
- signal conditioning and data conversion (analog-to-digital conversion or digital-to-analog conversion),
- AI-based intelligent sensor and actuator data processing algorithms for various intelligent capabilities,
- time-synchronization by an internal clock with optional external time references such as Global Positioning System (GPS), one pulse-per-second (1PPS), Inter-Range Instrumentation Group B (IRIG-B), IEEE 1588 Precision Time Protocol (PTP), or Network Time Protocol (NTP),
- secure network communication with the outside world through an IEEE 1451-based network communication module via different connectivity, including Ethernet/Internet, 4G/LTE/5G/6G networks.

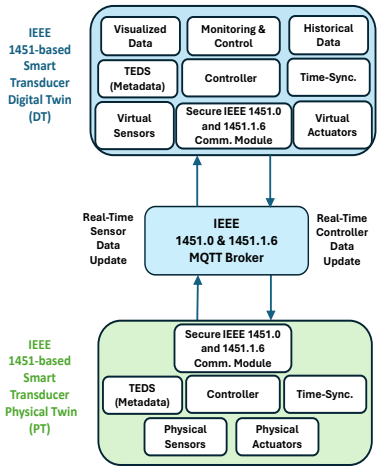


Fig. 2. IEEE 1451.0 and 1451.1.6-based Smart Transducer DT.

B. *IEEE 1451.0 and 1451.1.6-based Digital Twin*

Fig. 2 shows the IEEE 1451.0 and 1451.1.6-based smart transducer DT for real-time monitoring and control, where the bi-directional communications between PT and DT are based on IEEE 1451.0 and 1451.1.6 standards. The smart transducer PT consists of a set of physical sensors and actuators, a controller, metadata of the smart transducer, time-synchronization, and an IEEE 1451.0 and 1451.1.6 communication module. The smart transducer DT consists of a set of virtual sensors and actuators, a controller, metadata, visualized and historical data, real-time monitoring and control algorithms, time-synchronization, and an IEEE 1451.0 and 1451.1.6 communication module. The real-time data or information exchange between the PT and DT via their 1451 communication modules is based on the IEEE 1451.0 network services/messages and 1451.1.6 topics via the MQTT broker. The DT can monitor and control the PT based on real-time sensor data from the PT, process sensor data for decision making for control, and then control the physical twin based on controller data update via IEEE 1451.0 and 1451.1.6 MQTT standard protocols and interfaces. The DT also visualizes sensor data and stores historical sensor data.

IV. IEEE 1451.0 AND 1451.1.6-BASED SMART TRANSDUCER DT TESTBED

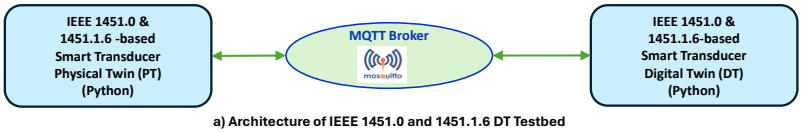
A. *IEEE 1451.0 and 1451.1.6-based Smart Transducer DT Testbed*

Fig. 3 shows an IEEE 1451.0 and 1451.1.6 -based smart transducer DT testbed for real-time monitoring and control. Fig. 3a) shows the architecture of DT testbed that consists of a smart transducer PT, a MQTT broker, and a smart transducer DT developed using Python. The real-time data or information exchange between the PT and DT is based on IEEE 1451.0 network service messages and 1451.1.6 topics using their 1451.0 and 1451.1.6 communication modules via the Mosquitto MQTT Broker **[16]. Fig. 3 b) shows a setup of DT testbed setup that consists of one smart transducer PT with a graphic user interface (GUI), one smart transducer DT with a GUI, and a Mosquitto MQTT broker running on a Linux laptop. A local area network (LAN) was created using a network switch connected between a Raspberry Pi [17] and a laptop running Ubuntu OS. The Raspberry Pi was used to run the scripts that control the smart sensor and actuator behavior, and the Ubuntu laptop was used to run the DT and the Mosquito MQTT broker.

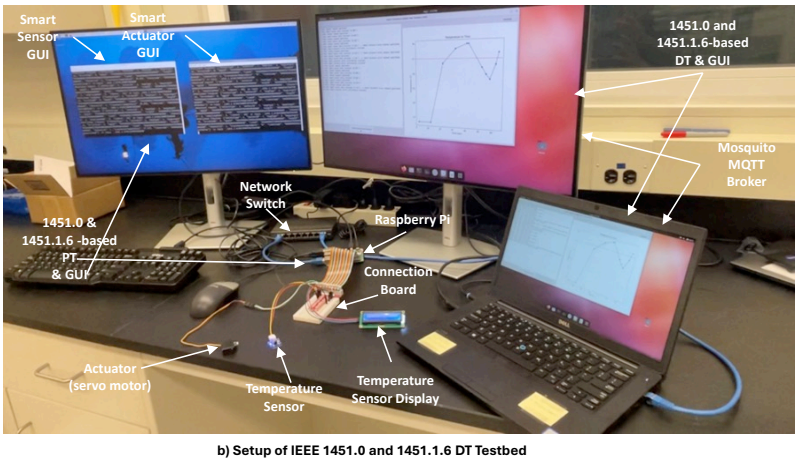
B. *Smart Transducer PT*

As shown in Fig. 3 b), the IEEE 1451.0 and 1451.1.6 smart transducer PT (consisting of a smart sensor PT and a smart actuator PT), has been developed based on the IEEE 1451.0 and 1451.1.6 standards using a Raspberry Pi and a sensor and actuator development kit [19] in the Python programming language. The smart transducer PT uses the Raspberry Pi (a micro-computer) to connect to both a temperature sensor and actuator (representing a smart home heating/cooling system) via a connection board.

- **Smart Sensor:** The smart sensor consists of a temperature sensor and a liquid-crystal display (LCD) screen that are connected to the Raspberry Pi via the connection board. The Smart Sensor (**`sensor_pubsub.py`**) was developed using Python. On invocation, the Smart Sensor will automatically connect to the MQTT broker running on the DT laptop via the LAN network switch, and subscribe to the IEEE 1451.0 messages and 1451.1.6 'SMARTSENSOR' topic. When it receives the IEEE 1451.0 read temperature sensor data Command message published by the DT, it will read the current temperature value from the temperature sensor based on the 1451.0 Command received, updates the sensor reading on the LCD screen with its current value, and then publish an IEEE 1451.0 Reply message with this sensor reading value to 'SMARTSENSOR'. The DT can subscribe to this topic to access this 1451.0 Reply message containing sensor reading value.



a) Architecture of IEEE 1451.0 and 1451.1.6 DT Testbed



b) Setup of IEEE 1451.0 and 1451.1.6 DT Testbed

Fig. 3. IEEE 1451.0 and 1451.1.6-based smart transducer DT testbed.

- **Smart Actuator:** The Smart Actuator consists of a servo motor that is connected to the Raspberry Pi via the connection board. The Smart Actuator (**`servo pubsub.py`**) is developed using Python. On invocation, the Smart Actuator will automatically connect to the MQTT broker running on the DT laptop via LAN, subscribe to the 1451.0 Command messages and 1451.1.6 'SMARTACTUATOR' topic. When it receives an IEEE 1451.0 write a data control Command message published by the DT, the servo motor will quickly turn to a set position based on the 1451.0 control message, then return to a neutral position after a fixed amount of time. This is meant to simulate toggling a device that changes/stabilizes the temperature. Lastly, it then publishes a 1451.0 Reply message to the 'SMARTACTUATOR' topic.

C. Smart Transducer DT

The smart transducer DT (**`pubsub.py`**) is developed using Python based on the IEEE 1451.0 and 1451.1.6 standards. On invocation, the DT (APP) will automatically connect to the MQTT broker, publish and subscribe to both the 'SMARTSENSOR' and 'SMARTACTUATOR' topics, and start continuously publishing 1451.0 Command messages to read sensor data to the 'SMARTSENSOR' topic every 10 seconds. When it receives a 1451.0 Reply sensor reading message from the Smart Sensor, it will check the temperature value, and if it is above/below certain temperature threshold (range 27± 2ºC), then it will publish a 1451.0 Command to write an actionable message to the 'SMARTACTUATOR' topic to turn on/off the Smart Actuator and control the temperature of a smart home. The DT shown in Fig. 3 b) shows a GUI for which the user can track published/received message events, view a line plot of all temperature data received by the Python DT during its runtime, and publish sensor data request messages on demand in real time.

D. Bi-Directional Communications between DT and PT Based on IEEE 1451.0 and 1451.1.6 Standards

The bi-directional communications between the DT and PT are based on IEEE 1451.0 network services/messages and 1451.1.6 topics using their 1451.0 and 1451.1.6 communication

```
sensorreq = {    # sensor data request
  "netSvcType" : 2,  # transducer access services
  "setSvcId" :   1,  # read from channel of TIM
  "msgType" :    1,  # command
  "msgLength" : 59,  # bytes
  "appId" :      UUID,
  "ncapId" :     UUID,
  "timId" :      UUID,
  "channelId" :  1,
  "samplingMode" :5,  # immediate
  "timeout" :    60
}
```

a) 1451.0 Command (read data) from DT to PT

```
sensorrep = {    # sensor read reply (data)
  "netSvcType" : 2, # transducer access services
  "netSvcId" :   1, # read from channel of TIM
  "msgType" :    2, # reply
  "msgLength" : 78, # bytes
  "errorCode" :  0,
  "appId" :      UUID,
  "ncapId" :     UUID,
  "timId" :      UUID,
  "channelId" :  1,
  "transducerSampleData" : str(read_temp()),
  "timestamp" :  60
}
```

b) 1451.0 Reply (read data) from PT to DT

```
servoreq = {     # actuator movement request
  "netSvcType": 2, # transducer access services
  "setSvcId":   7, # write to channel of TIM
  "msgType":    1, # command
  "msgLength":  75,# bytes
  "appId":      UUID,
  "ncapId":     UUID,
  "timId":      UUID,
  "channelId":  1,
  "samplingMode":5, # immediate
  "dataValue":  0, # 16 bytes string
  "timeout":    60,
}
```

c) 1451.0 Command (write data) from DT to PT

```
servorep = {     # actuator movement reply
  "netSvcType" : 2, # transducer access services
  "netSvcId" :   7, # write to channel of TIM
  "msgType" :    2, # reply
  "msgLength" : 52, # bytes
  "errorCode" : 0,
  "appId" :      UUID,
  "ncapId" :     UUID,
  "timId" :      UUID,
  "channelId" :  1
}
```

d) 1451.0 Reply (write data) from PT to DT

Fig. 4. IEEE 1451.0 Command and Reply messages.

modules via the MQTT Broker. DT can publish the IEEE 1451.0 command message to the PT via the MQTT broker. The PT can subscribe to the 1451.0 Command messages from the DT, and then publish the 1451.0 Reply message to the DT. And then, DT can subscribe to the 1451.0 Reply messages from the Smart Sensor and Smart Actuator PT, process sensor data, and make decisions for real-time monitoring and control.

Fig. 4 a) and Fig. 4 b) show IEEE 1451.0 Command and Reply message formats of read sensor data from smart sensor, respectively. The Command message consists of netSveType (2: transducer access service), netSveId (1: read data from one channel of one TIM), msgType (1:command), msgLength (59 bytes), appId (UUID, 16 bytes), ncapId (UUID, 16 bytes), timId (UUID, 16 bytes), channelId (two bytes, 1), samplingMode (1 byte, 5: immediately read), and timeout (8 bytes). The 1451.0 Reply message consists of netSveType (2), netSveId (1), msgType (2-Reply), msgLength (78 bytes), errorCode (1 byte, 0-no error), appId (UUID,16 bytes), ncapId (UUID,16 bytes), timId (UUID, 16 bytes), channelId (2 bytes, 1), transducerSmapleData (16 bytes, string), and timestamp (10 bytes). Fig. 4 c) and Fig. 4 d) show the Command and Reply message formats of write actuator (control) data to smart actuator, respectively. The Command message consists of netSveType (2: trasducer access service), netSveId (7: write data into one channel of one TIM), msgType (1: command), msgLength (75 bytes), appId (UUID, 16 bytes), ncapId (UUID, 16 bytes), timId (UUID, 16 bytes), channelId (2 bytes, 1), samplingMode (1 byte, 5: immediately write), dataValue (16 bytes, string), and timeout (8 bytes). The Reply message consists of netSveType (2), netSveId (7), msgType (2-Reply), msgLength (52 bytes), errorCode (2 bytes), appId (UUID, 16 bytes), ncapId (UUIID, 16 bytes), timId (UUID, 16 byte), and channelId (2 bytes, 1). IEEE 1451.1.6 topic format [6] is defined as "_1451.1.6/D0/***". In this case, Smart Sensor topic is defined as "1451.1.6/D0/SMARTSENSOR". Smart Actuator topic is defined as "1451.1.6/D0/SMARTACUTATOR".

Fig. 5: Sensor debug panel of receiving command and sending reply messages.

V. Test Case and Testing Results of IEEE 1451.0 and 1451.1.6 DT Testbed

A simple test case of real-time temperature monitoring and control of smart home was tested using IEEE 1451.0 and 1451.1.6 DT testbed. The DT continuously publishes the 1451.0 Commands to read sensor data from the Smart Sensor PT, the Smart Sensor PT can continuously subscribes to the 1451.0 Command messages, read temperature sensor data, and then publishes the Reply messages including temperature value to the DT via the MQTT broker. The DT continuously subscribes to the temperature values from the Smart Sensor PT, and checks if the temperature value is in the range of control thresholds (e.g., 27 ± 2 ºC). If the temperature value exceeds the maximum threshold, then the DT publishes a Command (control, write data) to the Smart Actuator PT, and when received, the Smart Actuator turns off the servo motor to stop heating and start cooling. If the temperature value is less than the minimum threshold, then the DT publishes a Command (write data) to the Smart Actuator PT, and when received the Smart Actuator turns on the servo motor to start heating and stop cooling.

Fig. 5 shows the smart sensor debug panel GUI that shows the communication process history to exchange the 1451.0 Command/Request and Reply/Response messages including different temperature values. The IEEE 1451.1.6 topic: "_1451.1.6/D0/SMARTSENSOR" is shown for SmartSensor in Fig. 5. The Command message (request) is shown in Fig. 5, which consists of netSveType (0x02, 2), netSveId (0x01, 1), msgType (0x01, 1-command), msgLength (0x00, 0x3b, 59), appId (16 bytes), ncapId (16 bytes), timId (16 bytes), channelId (0x01, 1), samplingMode (0x05, 5), and timeout (8 bytes). The Reply (response) message consists of netSveType (0x02, 2), netSveId (0x01, 1), msgType (0x02, 2-Reply), msgLength (0x00, 0x4e, 78 bytes), errorCode (0: no error), appId (16 bytes), ncapId (16 bytes), timId (16 bytes), channelId (0x01, 1), transducerSmapleData (16 bytes, 299.237 K, 26.937 C), and timeout (8 bytes). Fig. 5 also shows that Smart Sensor topic is "1451.1.6/D0/SMARTSENSOR".

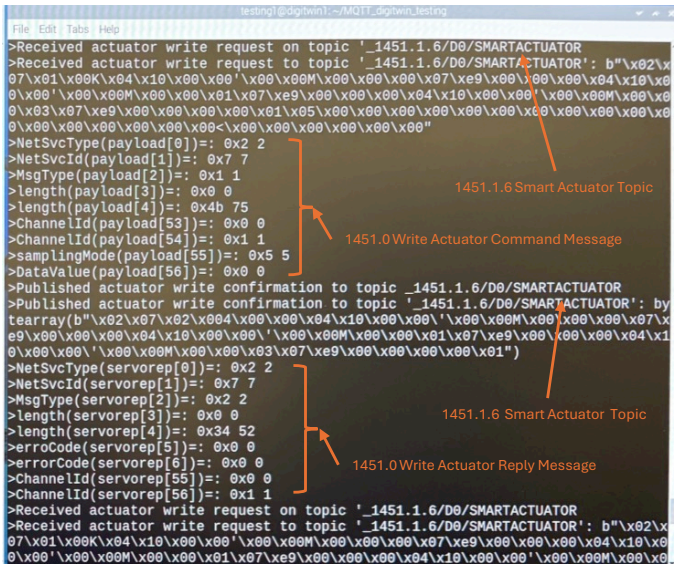
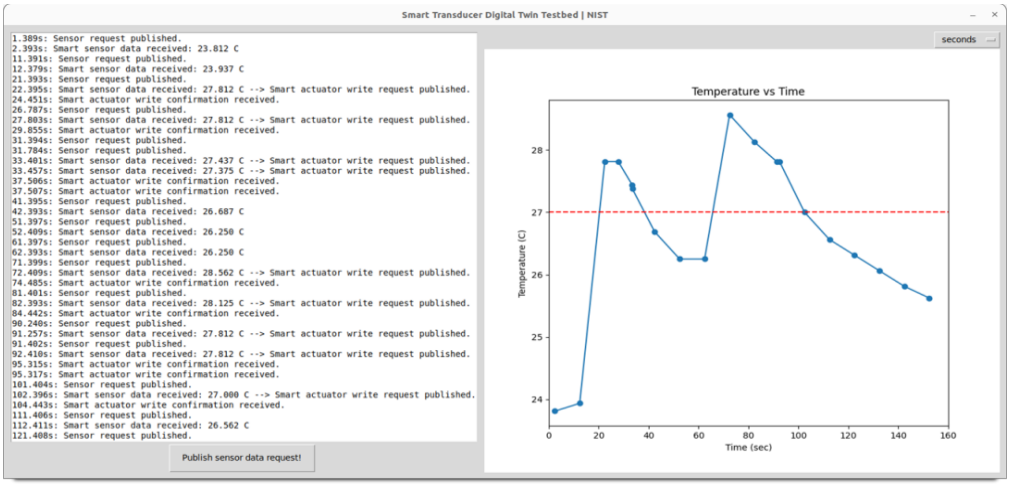


Fig. 6. Actuator servo debug panel of receiving command and sending reply messages.

Fig. 6 shows the smart actuator debug panel GUI that shows the communication process history of exchanges of the 1451.0 Command/Request and Reply/Response messages, which include Control Command (on/off, or 1/0). Fig. 6 show IEEE 1451.0 Command and Reply message formats of write actuator (control) data to smart actuator, respectively. The 1451.0 Command message consists of netSveType (0x02, 2), netSveId (0x07, 7- write data), msgType (0x01, 1-command), msgLength (0x00, 0x4c, 75), appId (16 bytes), ncapId (16 bytes), timId (16 byes), channelId (0x01, 1), samplingMode (0x05, 5), dataValue (0x00, 0), and timeout (8 bytes). The Reply message consists of netSveType (0x02, 2), netSveId (0x07, 7), msgType (0x02, 2-Reply), msgLength (0x00, 0x34, 52), errorCode (0x00, 0), appId (16 bytes), ncapId (16 bytes), timId (16 bytes), and channelId (0x01, 1). Fig. 6 also shows that Smart Actuator topic is "1451.1.6/D0/SMARTACUTATOR".



a)1451.0 Command and Reply messages b) Temperature sensor curve for monitoring and control

Fig. 7. Real-time monitoring and control testing results of smart transducer DT GUI.

Fig. 7 shows the testing results of real-time monitoring and control of smart homes using IEEE 1451.0 and 1451.1.6 smart transducer DT testbed. As shown in Fig. 7 a), the DT uses a GUI for users to track published/received message events, view a line plot of all temperature data received by the DT during its testing runtime, and publish sensor data request messages on demand in real time. Fig. 7 b) shows smart home temperature control in the range of minimum and maximum thresholds. While this study employed a simplified model for the experimental setup, the implementations of DT and PT based on the IEEE 1451.0 and 1451.1.6 standards, which offers several advantages, such as incorporating metadata related to sensor types and accuracy into the DT, achieve smart transducer data interoperability based on the IEEE 1451.0 message formats and 1451.1.6 topics. Consequently, the use of standardized technology facilitates the interoperable and scalable DT.

VI. Summary and Conclusion

This paper describes an IEEE 1451.0 and 1451.1.6-based smart transducer DT testbed for IoT real-time monitoring and control of smart homes. This testbed consists of an IEEE 1451.0 and 1451.1.6-based smart transducer (sensor and actuator) PT, its corresponding smart transducer DT, one MQTT broker, and a bidirectional information exchange between PT and DT for real-time monitoring and control. The DT and PT can communicate with each other via the MQTT broker using IEEE 1451.0 Command and Reply messages and 1451.1.6 topics for monitoring and controlling the temperature of an example IoT application system. The monitoring and control results are provided in the paper to test and validate that the IEEE 1451.0 and 1451.1.6-based PT and DT work correctly.

Future work will include adding multiple heterogeneous sensors and actuators into this DT testbed, implementing more 1451.0 network services/functions (e.g., read/write TEDS), adding more real-world test cases or realistic control scenarios, and enhancing GUI to display results. This testbed will be used to test, measure, analyze interoperability and performance (e.g., latency, packet loss rate, and throughput) of commercial IEEE 1451.0 and 1451.1.6-based smart transducers (sensors and/or actuators).

***NIST Disclaimer:** Certain commercial equipment, instruments, or materials, commercial or non-commercial, are identified in this paper in order to specify the experimental procedure adequately. Such identification does not imply recommendation or endorsement of any product or service by NIST, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.

References

[1] Foundational Research Gaps and Future Directions for Digital Twins, [Online]. Available: https://nap.nationalacademies.org/catalog/26894/foundational-research-gaps-and-future-directions-for-digital-twins

[2] IIC, Digital Twins for Industrial Applications, [Online]. Available: https://www.iiconsortium.org/pdf/IIC_Digital_Twins_Industrial_Apps_White_Paper_2020-02-18.pdf

[3] Top 8 Digital Twin Benefits, [Online]. Available: https://www.ptc.com/en/blogs/corporate/digital-twin-benefits.

[4] Decoding Digital Twins: Exploring the 6 main applications and their benefits, [Online]. Available: https://iot-analytics.com/6-main-digital-twin-applications-and-their-benefits/

[5] IEEE 1451.0-2024 Standard for a Smart Transducer Interface for Sensors and Actuators--Common Functions, Communication Protocols, and Transducer Electronic Data Sheet (TEDS) Formats, IEEE 1451.0-2024. [Online]. Available: *https://ieeexplore.ieee.org/document/10571828*.

[6] IEEE 1451.1.6-2025 Standard for a Smart Transducer Interface for Sensors, Actuators, and Devices - Message Queue Telemetry Transport (MQTT) for Networked Device Communication, [Online]. Available: https://store.accuristech.com/standards/ieee-1451-1-6-2025?product_id=2919634&srsltid=AfmBOopS9CxkqavzmKCSgIW8AWHEW0lrvee9PHid47MHiVmPyC_3ri5x

[7] E. Song, M. J. Burns, A. Pandey, T. P. Roth, IEEE 1451 Smart Sensor Digital Twin Federation for IoT/CPS Research, 2019 IEEE Sensors Applications Symposium (SAS), MAY 6, 2019, [Online]. Available: DOI:10.1109/SAS.2019.8706111.

[8] Da Rocha, H., Pereira, J., Abrishambaf, R.; Espirito Santo, A. An Interoperable Digital Twin with the IEEE 1451 Standards. Sensors, 2022, 22, 7590, [Online]. Available: https://doi.org/10.3390/s22197590.

[9] E. Song, Shanaka P, A., H. Nishi, T. Rooth, IEEE 1451-based Digital Twin Framework for Real-Time Monitoring and Control of Smart Homes, IECON 2025, Oct. 14-17, 2025, Madrid, Spain, [Online]. Available: DOI: 10.1109/IECON58223.2025.11221119.

[10] ISO 23247: Digital Twin Framework for Manufacturing, [Online]. Available: *https://ap238.org/iso23247/*

[11] ISO/IEC JTC 1, SC 41: Internet of Things and digital twin, [Online]. Available: https://jtc1info.org/sd-2-history/jtc1-subcommittees/sc-41/

[12] Digital Twin System Interoperability Framework, [Online]. Available: *https://www.digitaltwinconsortium.org/pdf/Digital-Twin-System-Interoperability-Framework-12072021.pdf*

[13] Digital Twin Testbed Program, [Online]. Available: , https://www.digitaltwinconsortium.org/initiatives/digital-twin-testbeds/.

[14] Twinsense Tetsbed: New testbed applies composability framework and AI evaluation, [Online]. Available: https://www.controleng.com/new-testbed-applies-composability-framework-and-ai-evaluation/

[15] E. Y. Song, G. J. FitzPatrick, and K. B. Lee, "Smart sensors and standard-based interoperability in smart grids," IEEE Sensors J., vol. 17, no. 23, pp. 7723–7730, Dec. 2017.

[16] Mosquito MQTT broker, [Online]. Available: https://mosquitto.org/.

[17] Raspberry Pi 4B Ultimate Kit, [Online]. Available: https://www.pishop.us/product/raspberry-pi-4b-ultimate-kit-64gb-sd-card/.

[18] Sensor Kit v4.0 for Raspberry Pi, [Online]. Available: https://www.pishop.us/product/sensor-kit-v4-0-for-raspberry-pi/.