International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 13 Issue: 11 | Nov 2026 www.irjet.net p-ISSN: 2395-0072

Digital Twin Technology for Enhanced Smart Building Energy Management

Prof. Priyanka R Navale[1], D Prarthana Kommineni[2], Priya Raju Tasildar[3], Sahana M Malakannavar[4], Sindhu P K[5]

[1]Asst. Professor, Department of Computer Science and Business Systems, BIET, Davangere-577004
[2-5]Student, Department of Computer Science and Business Systems, BIET, Davangere-577004

Abstract - *This paper presents a low-cost smart-home digital twin prototype intended to improve visibility, safety and energy awareness in a connected building environment. The physical model uses an Arduino Uno with sensors for temperature and humidity, ambient light, infrared presence, gas concentration, water presence, tank distance, load current and supply voltage. The controller operates a lamp, fan, water pump, buzzer, tank-refill relay and gate motor using automatic rules and user commands. Sensor readings are transmitted over USB serial to a Python Flask server, which converts the raw values into a structured virtual representation of the living room, kitchen, bedroom, tank, gate, power supply and battery. A Bootstrap dashboard refreshes sensor information every two seconds and provides device controls. The prototype also integrates YOLOv8n and OpenCV for person detection and colour-based delivery-category suggestions, together with a browser voice assistant activated by the wake word "Buddy". The implementation demonstrates synchronized monitoring and control using inexpensive hardware and open-source software. Energy values shown by the twin include derived estimates; the project does not report a validated building-scale energy-saving percentage or machine-learning energy forecast.*

Key Words: Digital Twin, Smart Home, Building Energy Management, IoT, Arduino Uno, Flask, YOLOv8, OpenCV, Energy Monitoring.

1. INTRODUCTION

Buildings and homes use energy for lighting, ventilation, appliances and other services. Conventional low-cost automation often provides individual appliance switching but does not present a unified view of environmental conditions, safety status and power usage. A connected system that monitors these parameters and represents them in a virtual model can help occupants understand the state of the physical environment and take timely action.

A digital twin is a virtual representation of a physical object or system that remains synchronized through data received from the physical environment. In the present project, the physical object is a table-top model home containing a living room, kitchen, bedroom, water tank, main gate and a 12 V battery-backed power supply. Its digital twin is implemented as a browser-based dashboard that mirrors the current state and allows the user to control selected devices through buttons or voice commands.

The project combines an Arduino-based sensor and actuator network, a Flask web application, a camera-based visitor detector and a browser voice assistant. The main focus of this paper is the digital twin architecture and its monitoring and energy-awareness functions.

2. RELATED WORK

The Internet of Things (IoT) connects physical objects equipped with sensing, computing and communication capabilities. A commonly used layered view contains a perception layer, network layer, processing or middleware layer and application layer [1], [2]. In this project, the Arduino and sensors form the perception layer, USB serial and the local network provide communication, Flask performs processing, and the browser dashboard forms the application layer.

The digital twin concept links a physical product, its virtual representation and the data connection between them [3]. A twin is more than a static display: it can organize live measurements, derive additional state information and support control. Web frameworks such as Flask provide a lightweight way to expose sensor values through REST-style endpoints, while OpenCV and YOLO support camera-based detection [4], [5].

3. PROBLEM STATEMENT

Many low-cost home automation systems provide isolated relay control without a consolidated representation of comfort, safety, water and power information. Some functions rely only on fixed switching, and the user may not know whether the physical system is operating normally. The project addresses this gap by designing a low-cost system that senses environmental and power parameters, applies automatic control rules and mirrors the system state in a browser-based digital twin.

4. OBJECTIVES

The objectives of the proposed work are:

1. To develop a digital twin model of the smart building environment.

2. To integrate IoT sensors for real-time monitoring of environmental conditions and energy consumption.
3. To automate and control electrical appliances using sensor data and user commands.
4. To develop a complete web-based system integrating real-time visualization, remote monitoring, device control, energy management, delivery person detection, safety alerts and voice assistant functionality.

5. SYSTEM ARCHITECTURE

The system follows a layered client-server architecture, shown in Fig -1. The Arduino Uno reads sensors and controls actuators. It communicates with a computer over USB serial at 9600 baud. A serial bridge stores the latest telemetry line in sensor_data.txt and forwards commands from command.txt to the controller. The Flask server reads the sensor file, validates the data, builds the digital twin state and writes command requests. The browser displays the dashboard, while a separate thread handles camera frames and YOLO-based person detection.

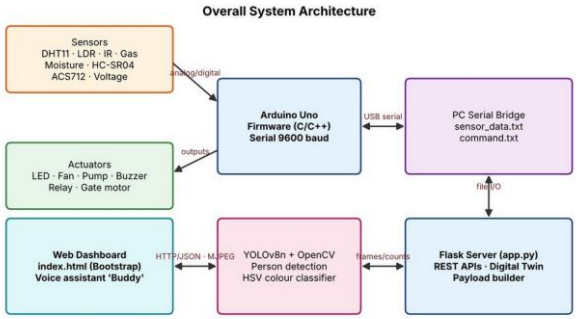

Fig -1: Overall system architecture of the digital twin prototype.

Two data paths are used. In the telemetry path, the Arduino sends a comma-separated line, the bridge stores it, Flask converts it into a structured payload, and the browser fetches the updated state. In the control path, a dashboard or voice command reaches a Flask endpoint, is mapped to a command character, and is forwarded to the Arduino through the bridge.

6. HARDWARE AND SOFTWARE COMPONENTS

The hardware includes an Arduino Uno (ATmega328P), DHT11 temperature and humidity sensor, LDR module, IR obstacle sensor, gas sensor, soil-moisture module, HC-SR04 ultrasonic sensor, ACS712 current sensor, voltage sensor module, relay, H-bridge motor driver, DC motor, fan, water pump, LED, buzzer and 12 V battery. A computer with a webcam runs the Flask server and YOLO model.

The software stack consists of Arduino IDE, Python 3, Flask, OpenCV, NumPy, Ultralytics YOLOv8, Bootstrap 5 and the browser Web Speech API. The server exposes REST endpoints for sensor state, power information, room details, serial diagnostics, detection data and device updates.

7. METHODOLOGY

7.1 Sensor Acquisition

Each sensor was first tested individually using the Arduino serial monitor. The firmware reads environmental, safety and electrical parameters and produces a line of twelve comma-separated values per cycle.

7.2 Control Logic

Automatic rules operate selected devices. The fan is switched on when temperature exceeds 37 °C or when the configured presence condition is met. The gas alarm is activated when the gas reading exceeds 800. The tank relay is controlled using ultrasonic distance, while pump operation is inhibited when the moisture sensor indicates water. Manual commands are also accepted over serial communication.

7.3 Digital Twin Generation

Flask validates the latest sensor line and converts raw values into a structured dictionary containing living_room, kitchen, bedroom, tank, gate, power, battery and house objects. If valid hardware data are unavailable, the server uses simulated values and reports the data status.

7.4 Dashboard and Interaction

A Bootstrap dashboard presents room conditions, occupancy, device states, power and battery information. Sensor data are refreshed every two seconds and detection data every 1.5 seconds. The user can issue commands through dashboard controls or the browser voice assistant.

7.5 Visitor Detection

A separate camera thread uses the YOLOv8n model to detect persons. OpenCV converts image regions to HSV colour space and classifies shirt and trouser colour ranges into Zomato, Swiggy and Postman-style categories. Alerts are generated for newly detected persons and delivery categories. These labels are colour-based suggestions, not identity verification.

8. DIGITAL TWIN MODEL AND ENERGY MONITORING

The digital twin does not simply copy the raw sensor values. It derives additional values for display. A single

measured temperature is used to estimate room temperatures through fixed offsets, while humidity values are estimated using fixed relationships. The tank percentage is calculated from ultrasonic distance, treating 10 cm as empty and 2 cm as full. Battery percentage is estimated from voltage between 11.8 V and 12.7 V, and charging status is inferred from current direction and magnitude.

The power panel displays voltage, current and calculated power. The project also displays energy-today and energy-month fields and estimates room-level power contributions using fixed values associated with device states. These are model-derived values in the prototype and should not be interpreted as calibrated building energy-meter measurements.

Table -1: Derived quantities used by the digital twin

Quantity	Rule / interpretation
Tank percentage	(10 − d) / (10 − 2) × 100; distance limited to 2–10 cm
Battery percentage	(V − 11.8) / (12.7 − 11.8) × 100; limited to 0–100%
Battery status	Charging if current < −50 mA; discharging if > 50 mA; otherwise idle
Gas level	Danger above 800; warning above 500; otherwise safe
Temperature zone	Hot at 30 °C or above; cold at 18 °C or below; otherwise normal
Occupancy	Derived from the IR sensor presence state

9. IMPLEMENTATION AND TESTING

The implementation was carried out incrementally. Sensors were checked individually, actuators and automatic rules were added, and the serial telemetry format was stabilized. The Flask server was then connected to the data file and REST endpoints. The dashboard was developed and linked to the endpoints through periodic polling. The YOLO detection thread and voice assistant were integrated after the main monitoring and control functions.

Testing was performed at unit, module, integration and system levels. Hardware tests covered temperature response, darkness detection, presence sensing, gas alarm, pump protection, tank refill, gate movement, current and voltage readings, and telemetry formatting. Software tests covered missing sensor files, stale data, malformed telemetry, API payload generation, unknown room requests and dashboard control commands.

The integrated prototype performs sensing, automatic control, remote control, visual classification and voice interaction as designed. Telemetry is generated approximately once per second; the dashboard refreshes sensor values every two seconds and detection results every 1.5 seconds. No measured percentage of energy reduction or validated energy-prediction model is available, so such numerical claims are not made here.

10. RESULTS AND DISCUSSION

The prototype demonstrates that a low-cost microcontroller can support multiple sensing and actuation functions when heavier processing is handled by a computer. The separation between Arduino firmware and the Flask server allows the controller to remain focused on sensor acquisition and device rules, while the server builds the virtual state and provides the web interface. Simulation mode and stale-data detection allow the dashboard to remain available when hardware data are absent or outdated.

Several limitations affect interpretation and scaling. Room temperatures, humidity and power values are derived from fixed rules rather than measured independently in every room. A single lamp, fan and pump are shared across the room cards. The dashboard has no user authentication and is intended for a trusted local network. Colour-based visitor classification may be affected by clothing and lighting conditions, and browser speech recognition depends on supported browsers and network availability.

11. ADVANTAGES AND APPLICATIONS

The system is low cost and uses commodity modules and open-source software. It provides a consolidated view of comfort, safety, water and power information; supports automatic rules with safety interlocks; offers dashboard and voice interaction; and can be accessed from browsers on the local network. Potential applications include smart homes and apartments, assistance for elderly or physically challenged users, offices and gated communities, educational laboratories, small farms and prototype testbeds for building digital twins.

12. LIMITATIONS AND FUTURE WORK

The current prototype is a table-top home model rather than a deployed commercial building system. It does not include cloud storage, user authentication or a dedicated mobile application. Per-room environmental sensing and independent appliance-level metering are not implemented. Future work includes adding wireless sensor nodes, database and cloud services, authentication, HTTPS, network isolation, more per-room actuators and calibrated energy meters. The digital twin can also be extended with detailed building models and validated energy analytics when suitable building data become available.

13. CONCLUSION

This paper presents a digital-twin-based smart-home prototype that integrates an Arduino sensor and actuator network with a Flask web application, a browser dashboard, YOLOv8-based person detection and a voice assistant. The system converts live sensor readings into a structured virtual representation of rooms, tank, gate, power and battery, and supports automatic as well as user-driven control. The implementation demonstrates a low-cost and modular approach to monitoring and interaction. Its energy information is partly derived from fixed rules; therefore, deployment in a full smart building and measurement-based energy optimization remain important directions for future work.

REFERENCES

[1] L. Atzori, A. Iera and G. Morabito, "The Internet of Things: A survey," Computer Networks, vol. 54, no. 15, pp. 2787–2805, 2010.

[2] J. Gubbi, R. Buyya, S. Marusic and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," Future Generation Computer Systems, vol. 29, no. 7, pp. 1645–1660, 2013.

[3] M. Grieves, "Digital Twin: Manufacturing Excellence through Virtual Factory Replication," 2014.

[4] M. Grinberg, Flask Web Development: Developing Web Applications with Python, 2nd ed., O'Reilly Media, 2018.

[5] G. Jocher et al., "Ultralytics YOLOv8," Ultralytics, 2023. [Online]. Available: https://github.com/ultralytics/ultralytics