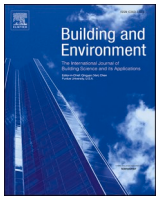
Building and Environment 295 (2026) 114420

Contents lists available at ScienceDirect

Building and Environment

journal homepage: www.elsevier.com/locate/buildenv



Development of microservices with machine learning algorithms for natural ventilation control in smart buildings

Wei Zhang [a], Leslie Norford [b], Wentao Wu [c,*]

[a] Graduate School of Design, Harvard University, Cambridge, MA 02138, United States
[b] Department of Architecture, Massachusetts Institute of Technology, Cambridge, MA 02139, United States
[c] School of Built Environment, Massey University, Auckland 0632, New Zealand

ARTICLE INFO

Keywords:
Microservices
Natural ventilation
Weather forecasting
Internet of Things
Smart building
Thermally activated building system

ABSTRACT

Smart buildings often struggle with the automatic control of complex heating, ventilation, and air conditioning systems, especially natural ventilation control. This paper introduces a novel microservices architecture to enable machine learning (ML) algorithms for natural ventilation control experiments in smart buildings. Implemented and evaluated in a three-story smart building in Cambridge, MA, from 2019 to 2021, the architecture incorporates a Python-based IoT network API and a Weather Forecast API. Experimental research demonstrated that predictive and reinforcement learning algorithms effectively controlled natural ventilation, optimizing CO_2 levels (800–900 ppm) and indoor air temperature (below 26 °C). Additionally, augmented TABS control, leveraging solar radiation prediction, successfully prevented overheating and saved heating energy. This study highlights the critical importance of microservices architecture in transforming complex building systems into scalable, resilient IoT frameworks for control research, enabling advanced ML for more climate-responsive and energy-efficient buildings.

1. Introduction

Artificial intelligence (AI) enables smart building systems to learn and adapt to the built environment using data from the Internet of Things (IoT). Advanced control and AI-based strategies including model predictive control (MPC) and reinforcement learning (RL) are increasingly applied in smart buildings to improve energy efficiency, thermal comfort, operational flexibility, and indoor air quality through optimization. Most existing studies focus on mechanical HVAC systems, where centralized control and well-defined actuation pathways facilitate automated and learning based control. For instance, Hari et al. reviewed over 200 studies on smart building and highlighted the substantial positive impact of machine learning on energy efficiency of building mechanical systems [1]. Similarly, Liu et al. examined more than 150 papers, confirming that machine learning improves advanced control systems in smart buildings [2]. Diego et al. [3] analyzed 174 manuscripts and concluded that the adaptive control system with AI algorithm is promising for controlling smart building mechanical systems.

Natural ventilation has the potential to reduce cooling energy consumption in urban smart buildings by 40 to 50 percent in Europe and North America [4], yet research on its control strategy remains comparatively limited despite its low-energy and climate responsive nature. Recent review studies indicate that natural ventilation control represents only a small fraction of the broader intelligent building control literature, which continues to be dominated by research on mechanical HVAC control approaches. For example, Mortari et al. [5] reviewed 44 studies on smart ventilation control strategies published between 2017 and 2023, of which only three [6–8] addressed natural ventilation control. Similarly, Rizo Maestre et al. [9] reviewed 51 studies on intelligent ventilation to improve indoor air quality published between 2017 and 2025, and only three [10–12] considered natural ventilation control.

Among existing studies on natural ventilation control, including those integrating with other building systems such as thermally activated building systems (TABS), the predominant method is simulation, whereas experimental validation in real buildings remains limited. In the two aforementioned reviews [5,9], four [6–8,11] of the six studies focusing on natural ventilation control relied primarily on simulation. For example, Chen et al. [11] proposed an ensembled-LSTM model to improve the MPC for natural ventilation, and the resulting MPC reduced significantly (86 %) the unmet CO_2 hours compared to rule-based control. The MPC is implemented in the whole building energy simulation

* Corresponding author.
E-mail address: w.wu2@massey.ac.nz (W. Wu).

https://doi.org/10.1016/j.buildenv.2026.114420
Received 24 September 2025; Received in revised form 14 February 2026; Accepted 25 February 2026
Available online 27 February 2026

program - EnergyPlus. Yoon et al. [14] analyzed the external and internal thermal mass of buildings with natural ventilation and concluded that natural ventilation can offset cooling demand on cooler days. The study is based on a developed analytical model. Han et al. [15] proposed a model-free reinforcement learning approach with transfer learning to control the slow-response radiant floor heating (RFH) system and reduced the heating demand by 15–23 % relative to rule-based control. While these studies represent significant advances in data driven multi-system building control, including natural ventilation and TABS, their future validation in real smart building environments supported by appropriate sensing and software architectures would substantially strengthen their practical relevance and impact. The scarcity of real world experimentation is further underscored by a broader review of natural ventilation in sustainable architecture, which analyzed 250 related publications [13]. Among the 12 papers categorized under emerging trends and AI driven real time control, only one study [12] reported a natural ventilation control experiment conducted in an operational smart building.

Simulation plays a central role in the design and preliminary evaluation of natural ventilation control algorithms by enabling systematic testing of control logic and sensitivity analyses across weather conditions, building parameters, and control settings. However, translating validated algorithms in virtual simulations to real built environment remains challenging, as local microclimatic effects, stochastic meteorological variability [14,15], and the detailed geometry of buildings and operable windows [16,17] governing airflow paths are often simplified or inadequately represented in simulation environments. These discrepancies can substantially affect the robustness and reliability of the control algorithms. Moreover, validation in real buildings is inherently difficult because, unlike centralized HVAC systems designed for automated control, natural ventilation relies on distributed components such as operable windows, vents, and façade elements and is tightly coupled with outdoor environmental conditions, requiring smart building infrastructures that support flexible sensing, real time data exchange, and sustained actuation under diverse operational and climatic conditions.

Existing smart building architectures are predominantly developed within an Internet of Things (IoT) paradigm that emphasizes large-scale sensing, connectivity, and data driven monitoring of building systems [18]. These frameworks provide effective support for real time data acquisition, fault detection, and supervisory control, and form the technological foundation of many commercial platforms. In the US HVAC industry, the BACnet protocol (ASHRAE 135 2024) further standardizes this paradigm by defining network topology, communication layers, and object models for building automation, while Annex W introduces RESTful web service interfaces and security mechanisms based on OAuth 2.0 [19]. In practice, BACnet based commercial APIs represent devices as objects and support data access and command execution [20], but long term control through web APIs is generally discouraged due to safety and security concerns [21]. As a result, most commercial smart building systems embed preprogrammed control strategies designed to ensure safe and reliable operation, with only limited and temporary access to actuators permitted. While sufficient for routine building management, this architecture does not support control research that requires sustained actuator authority, frequent control logic updates, and exposure to diverse operational and environmental conditions, which are essential for developing, validating, and refining learning-based natural ventilation control strategies.

Microservice architecture is an API-based, open, and evolutionary software paradigm characterized by loose coupling, modular deployment, and well-defined interfaces, making it particularly well suited to smart building automation. By decomposing monolithic systems into distributed, single purpose services modeled around specific functional domains [22–25], microservices decouple sensing, control, and actuation and allow advanced control algorithms to be introduced, modified, or withdrawn without altering the underlying commercial building infrastructure. According to Gartner's survey [26], 223 out of 300 global IT engineering organizations have adopted microservices architecture in information product development in 2023. Microservices have been used in smart buildings to transform monolithic IoT architectures into scalable and resilient systems [27–29], support framework based designs such as the Zachman architecture [30], and orchestrate cloud edge computing platforms through standardized API services [27,31]. However, none of those microservice architectures explicitly addressed natural ventilation control.

This study presents a case study to enable real-world control experiments, including natural ventilation and TABS. A microservice-based control framework is developed and implemented as a loosely coupled extension to the existing smart building architecture. Rather than modifying the vendor-provided control system, the proposed framework operates as a plug-in layer that supports sustained actuator access, modular control deployment, and seamless switching between normal operation and experimental control. Through this case study, the paper demonstrates a practical and collaborative pathway for bridging the gap between simulation-based natural ventilation control and validation in real buildings. This paper is structured in five sections. Section 2 outlines the microservices architecture in this research. Section 3 details the adaptation of microservices architecture in a smart building. Section 4 presents the control experiments implemented in a smart building with the microservices architecture. Finally, Section 5 summarizes the conclusions derived from this research.

2. Microservices architecture design

2.1. Microservices architecture

The microservices architecture proposed in this study enables the deployment of machine learning algorithms primarily for the control of natural ventilation (NV) [32] in a case-study building. As the building is equipped with TABS [33], the architecture also supports TABS operation as well as the coupled control of NV and TABS. The main functions of microservices include: 1) effectively connecting all the devices, including sensors and actuators, transforming them into accessible IoT objects, and providing a user-friendly interface for machine learning and optimization algorithms programming; and 2) collecting weather forecasting information from public service APIs, realizing calculations related to cloud conditions and solar irradiation, and enhancing the data feasibility for algorithms programming. Both functions would extend the existing IoT network and enable predictive ability in smart buildings. Each function is realized as an independent microservice (Fig. 1).

2.2. Microservice 1: IoT network API

The IoT network API is a Python-based library with a four-layer structure. The Private Supportive Function layer wraps the communication details in HTTP/RESTFul protocol. The Fundamental APIs layer defines basic API functions such as retrieving communication tokens and interaction with sensors and actuators. The Application APIs layer defines the control action for different actuators (window, sun shading, valve). All the IoT sensors and actuators are modeled as Object-oriented programming (OOP) objects at this layer. The Physical Phenomenon APIs layer defines complex operations, such as single-sided ventilation, cross-ventilation, building isolation, etc. The customized APIs, such as historical sensor data or actuator status retrieval API, are also designed at this level (Fig. 2, Appendix A-Table 4). The design philosophy is as follows: all the IoT sensors and actuators should be easily used in algorithm programming. The communication inside the four-layer structure architecture is transparent for programming. The token management should be simple to keep programming flexible.

2.3. Microservice 2: weather forecast API

The weather forecast API augments the available information from

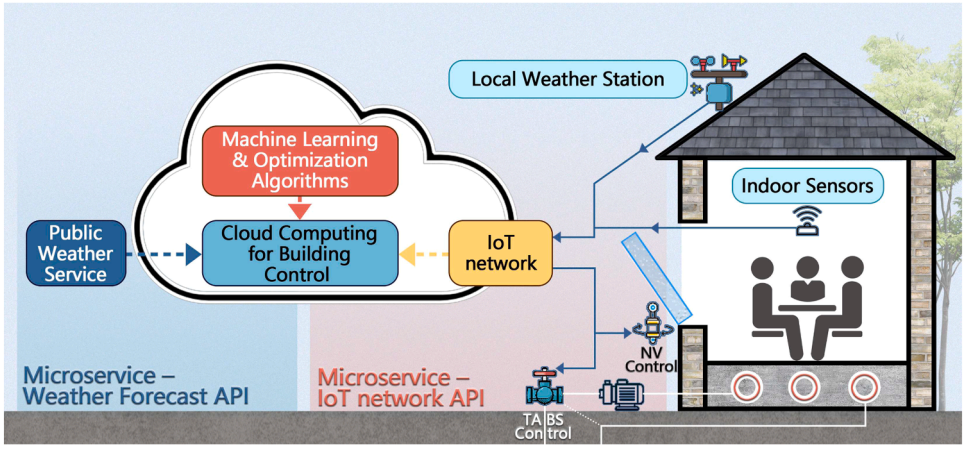


Fig. 1. Building components and the microservices.

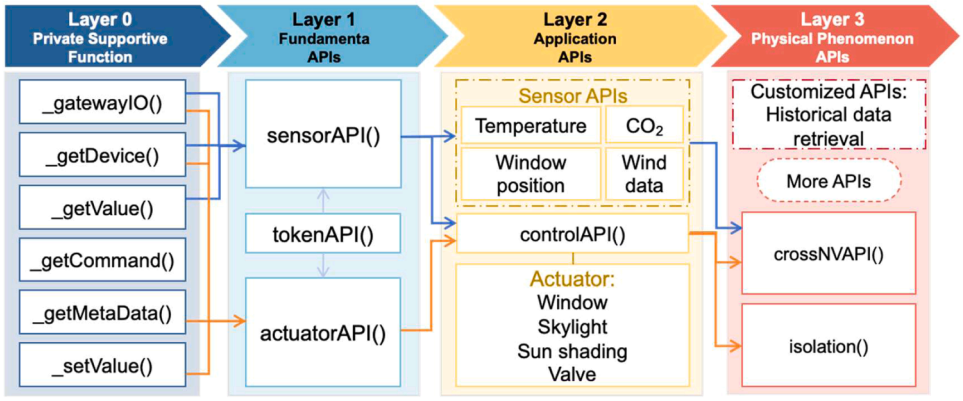


Fig. 2. IoT network API.

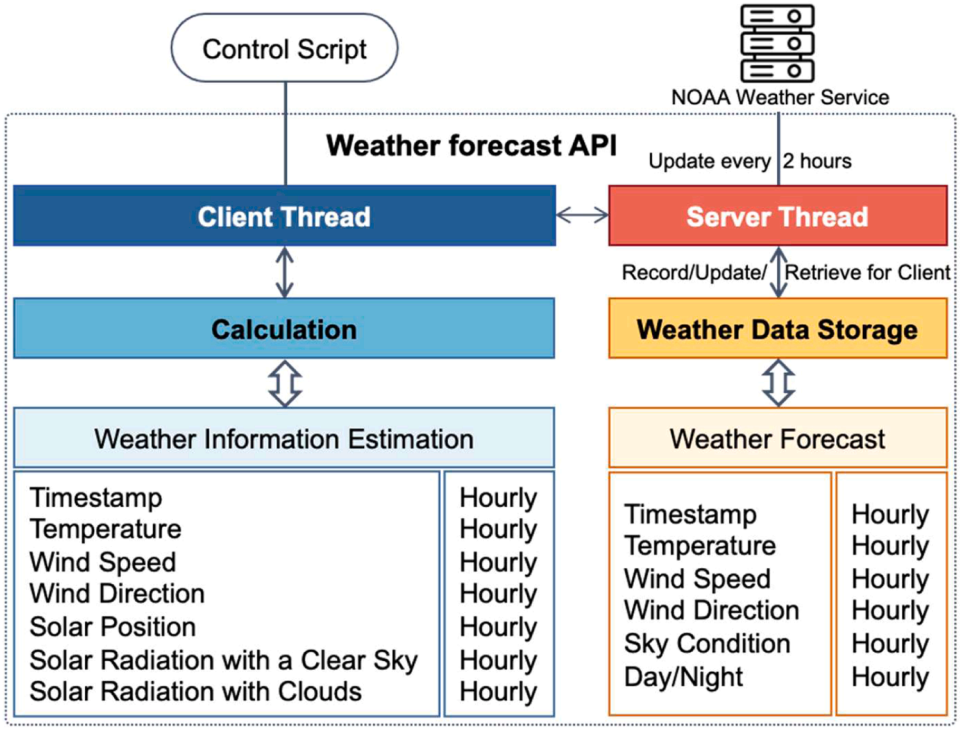


Fig. 3. Weather forecast API.

the built environment. With the geographical information of a specified building, it calculates the localized weather forecast information in the cloud. This API is designed in server/client structure: the server regularly retrieves the basic weather information from the National Oceanic and Atmospheric Administration (NOAA) public weather service. A copy of this information is stored in the cloud server and updated at a fixed frequency if the NOAA weather service is available; otherwise, the locally stored weather forecast information will not be renewed. The storage in the cloud increases the stability of information. When the control algorithm requests the weather forecast information, a client thread is created to fetch the information from the server thread, estimate the real solar radiation at 15 min intervals, and respond to the control script (Fig. 3). The details of solar radiation estimation is avaialable in Appendix A.

3. Implementation of the microservices architecture

The proposed microservices architecture was developed, installed, and evaluated in a real building from 2019 to 2021. This section presents the general building systems and IoT framework in the experimental building. Data was collected during 2019–2021 and is published in the work of Zhang [34] and Zhang et al. [12].

3.1. Brief of the case-study building

The three-story building [35] used to implement the microservices architecture has a natural ventilation system and a TABS. More detailed information about the building and building system is accessible in published literature [33,36]. The south office on the 3rd floor was designed as a configurable laboratory (hereafter lab). The dimensions of the lab are 3.85 m in depth, 2.6 m in width, and 3 m in height. The lab consists of three motorized windows for natural ventilation. Two of them are upper windows on the south wall, with a maximum opening of 18 degrees. The TABS components are installed in the technical closet. The lab includes sensors to measure indoor air temperature and CO_2 concentration, sensors to monitor the window opening angles and valve angles for controlling flow for TABS, and temperature sensors embedded in the concrete slab. All sensors are connected to the servers (Fig. 4). Although the building has more than 300 sensors and meters connected to the IoT network and the lab is equipped with a wall-mounted air sensor, an occupancy sensor and an embedded slab temperature sensor [36], additional measurements are still necessary for control model development and building monitoring. For example, in the lab, portable air sensors measure additional spatial air temperatures and CO_2 concentrations at different heights. The portable surface sensors measure the slab surface temperatures. These portable sensors are not integrated into the building's IoT network, and their measurement data must be obtained through their respective data-logging systems (Table 2). Although integrating additional sensors into the IoT network is technically feasible, it can substantially increase equipment and installation costs. Furthermore, portable sensors are not typically provided by smart-building vendors, limiting their immediate availability and making integration more complex. A second weather station in a nearby building is also used to supplement environmental data as reference micro-climate information [37].

3.2. ICT framework adaptation with microservices for case-study building

The necessary adaptation and design at the Information and Communication Technology (ICT) level was realized in the smart building IoT network to apply the microservices architecture in the smart building, especially in the lab for experiments. During 2019–2021, the existing building energy management system [33] is summarized as a dual-server architecture in section 3.2.3 of [34]. A dedicated hardware server controls the NV and TABS because of their crucial function in building operation; a centralized building energy management system (EMS) controls all the other building systems, including the geothermal heat pump (HP) and lighting. As a prerequisite of machine learning algorithm control, a third cloud server is deployed and transfers the existing IoT architecture into a Tri-server Architecture. With the necessary re-configuration of three servers, the microservice 1 - IoT network API maps all the IoT sensors and devices as classes and objects based on the OOP paradigm. The microservice 2 - weather forecast API,

Table 2
Portable sensor list.

Portable sensors	Manufacturer & Type	Function	Data logging
Room air sensor	Netatmo	Room air temperature, CO_2 level, humidity, air pressure, noise	Online logging, sensors are registered in campus Wi-Fi network
Surface temperature sensor	HOBO TMC6HE	Slab, roof, and wall surface temperature	Storage in device
Gund Hall weather station [37]	HOBO U30	Outdoor air temperature, wind speed and direction, solar radiation	Online logging

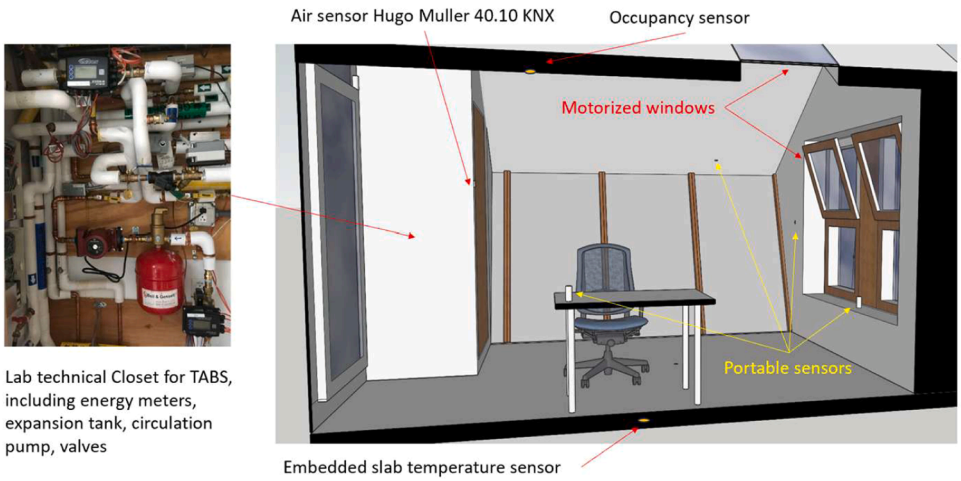


Fig. 4. The lab on the 3rd floor.

which calculates and stores the weather forecasting information, such as solar position and real solar radiation intensity with clouds, enables the weather prediction in this IoT architecture. The ICT level adaptation for microservices architecture is summarized in Fig. 5, and its engineering details are available in Appendix B. Rather than representing a conceptual network design centered on topology, the Tri-server Architecture emerges from an ICT adaptation based on microservices for the case-study building. It is implemented on top of the existing vendor-integrated network infrastructure of the building, leveraging established systems rather than introducing a fundamentally new network topology.

The hardware management server adopts the KNX protocol to interface with NV related actuators and sensors and executes field-level control. The API service management server uses the BACnet protocol, with interoperability between the two servers achieved through a KNX BACnet gateway. Both servers are deployed on the Microsoft Windows operating system and are equipped with commercial building management system software provided by commercial vendors. The building management system on the API service management server supports OAuth 2.0 based web services and API services, enabling secure communication and system integration. The computing server is a cloud based server that hosts all microservices and advanced control algorithms, including the training and deployment of machine learning and reinforcement learning based controllers. Researchers can remotely access the computing server through the Internet.

3.3. Microservices application

The usage of microservice 1 - IoT network API is illustrated with two programming examples (Appendix - Fig. 14). One script (left) that retrieves outdoor and room air temperature data and activates the south window at a predefined threshold, and another script (right) that performs simple CO_2 demand control by opening the operable window when indoor CO_2 concentration exceeds a specified level.

In the first script, a token is retrieved from the API server. Immediately, this token is used with two sensor APIs for the outdoor air temperature sensor (outdoorT) and lab room air temperature sensor (z31roomT). Then it is used with the controlAPI to open the south window in the lab for 20 %. The output shows that the sensor readings are retrieved as 25.9 °C for outdoor air temperature and 24.3 °C for lab room air temperature. The south window progressively opens, with three positions reading 0, 7 %, and 16 %. The actuator position checking number is adjustable; it's set as 3 to show the actuator position in the examples after the command is sent. For information security, a token has a limited life. The token management strategy is designed to extend its life by reusing it during control intervals. In the second script, a simple CO_2 demand control for 10 h is demonstrated by the token management strategy. In each control interval of 5 min, the token is used with sensor reading. If the CO_2 sensor reading is larger than 1000 ppm, the south window is open at 100 %; if the CO_2 sensor reading is less than 800 ppm, the south window is closed. This CO_2 demand control script is used to generate data for NV system identification [38]. Fig. 6 demonstrates the data generation on Feb 10, 2020, when the outdoor air temperature is between 7 °C and 7.5 °C.

The usage of microservice 2 - weather forecast API is illustrated in Fig. 7 and Table 3. The server thread retrieves and stores forecasted values of temperature, wind speed, wind direction, sky condition, and daytime from the National Weather Service public API. Once the client

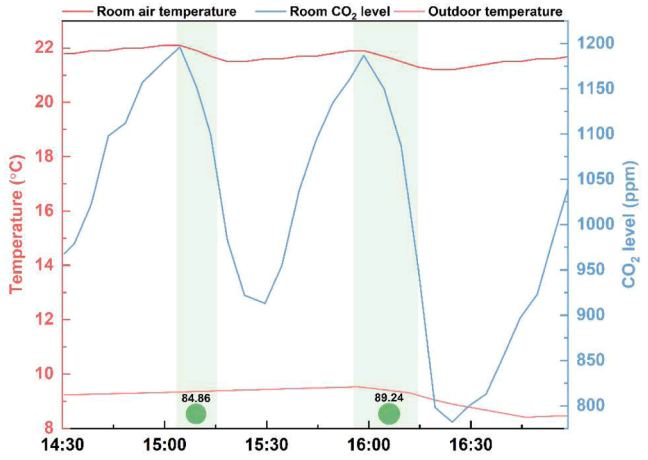


Fig. 6. Natural ventilation data generation using the CO_2 demand control script. Shaded area represents window opening. The bubbles represent the average opening angle during the shaded period.

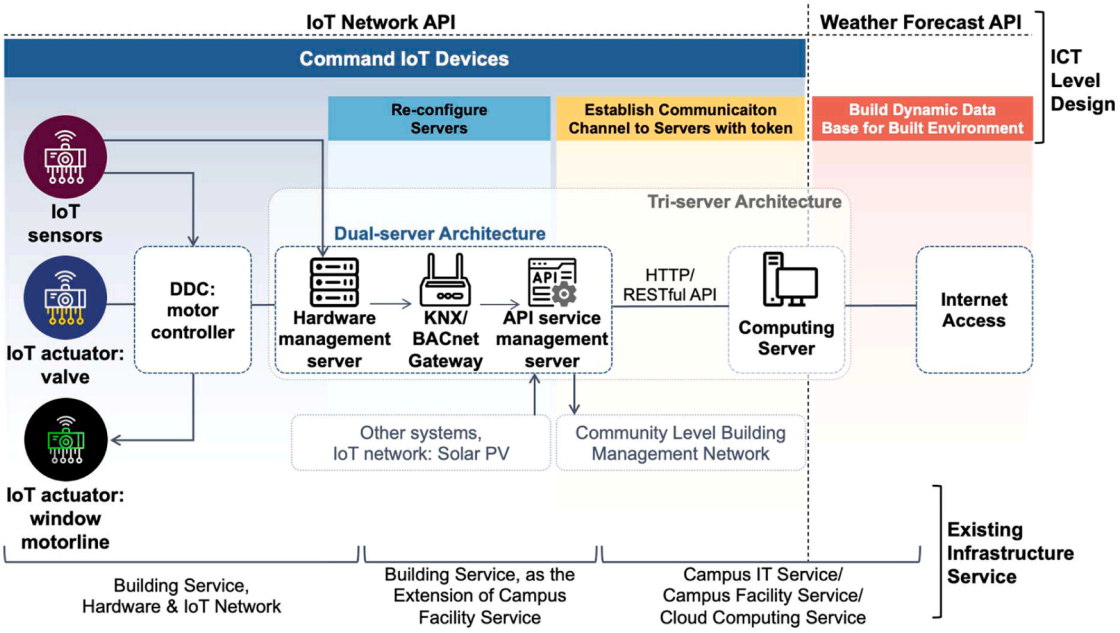


Fig. 5. ICT level adaptation 2019 – 2021: Tri-server architecture.

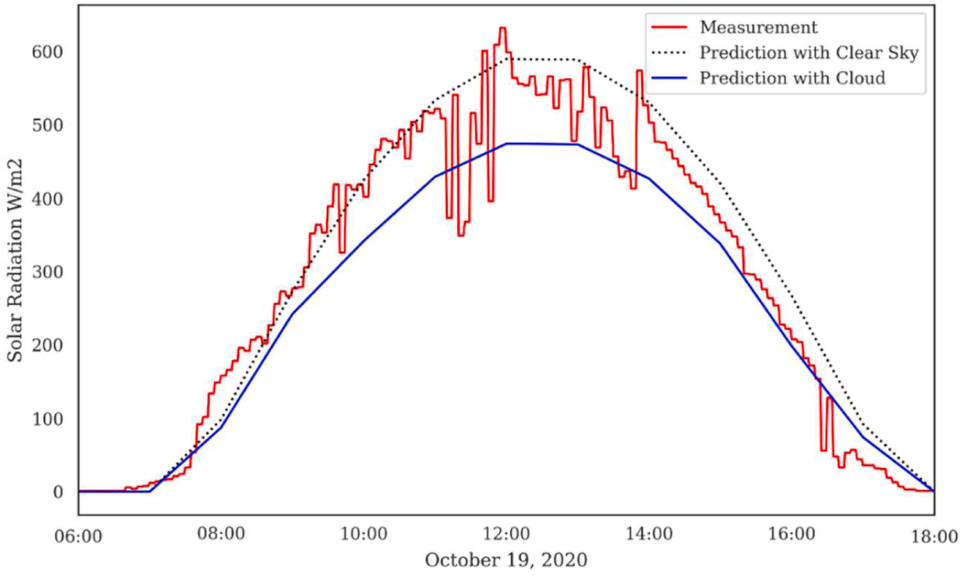


Fig. 7. Comparison of solar radiation prediction and measurement.

Table 3
Calculation of weather forecast API - server thread.

date_time	temperature	windspeed	direction	sky_condition	daytime	zenith	azimuth	ClearSky Solar	Cloud	CloudSolar
2020–10–30 06:00:00	3.33	0.45	0	Rain Showers	1	104.2	95.47	0	0.625	0
2020–10–30 07:00:00	2.78	0.45	0	Rain Showers	1	93.32	105.31	0	0.625	0
2020–10–30 08:00:00	2.22	0.45	0	Snow Showers	1	82.96	115.59	57.4	0.625	42.6
2020–10–30 09:00:00	1.67	0.45	0	Snow Showers	1	73.5	126.94	220.7	0.625	164
2020–10–30 10:00:00	1.67	0.45	0	Snow Showers Likely	1	65.46	139.95	367.9	0.625	273.4
2020–10–30 11:00:00	0.56	0.45	0	Light Snow Likely	1	59.51	154.97	473.6	0	473.6
2020–10–30 12:00:00	0.56	0.45	0	Snow Likely	1	56.34	171.79	527.7	0	527.7
2020–10–30 13:00:00	1.11	0.45	0	Chance Light Snow	1	56.46	189.31	525.7	0	525.7
2020–10–30 14:00:00	1.11	0.45	0	Chance Snow	1	59.85	206.02	467.7	0	467.7
2020–10–30 15:00:00	1.67	0.45	0	Slight Chance Light Snow	1	65.98	220.9	358.6	0	358.6
2020–10–30 16:00:00	2.22	0.45	0	Slight Chance Light Snow	1	74.14	233.76	209	0	209
2020–10–30 17:00:00	1.67	4.02	0	Cloudy	1	83.68	245	46.9	1	14.4
2020–10–30 18:00:00	1.11	3,58	0	Mostly Cloudy	0	94.1	255.21	0	0.625	0
2020–10–30 19:00:00	1.11	3.58	0	Partly Cloudy	0	105	265.02	0	0.375	0

thread requests, the solar zenith angle, solar azimuth angle, and solar radiation with clear sky conditions are calculated through Eqs. (A5)-(A8) (Appendix A). The solar radiation prediction with cloud amount is calculated by Eq. (A9) (Appendix A). Fig. 7 compares the solar radiation prediction and measurement from the Harvard Gund Hall weather station in Table 2. All the hourly information in Table 3 is available in the

Table 4
Application of software development principles.

Principles	Description	Implementation in microservices
Single Responsibility	Each function has only one responsibility	Microservices approach: Two independent APIs (IoT network API and weather forecast API) IoT network API: Each API function at the level 0 and 1 has only one function.
Open/Closed	The API is open for extension but closed for modification	IoT network API: In the case of adding a new IoT advice API at level 2, the API at level 1 can be extended without any modification for the existing API at level 2.
Liskov's Substitution	Derived APIs must be substitutable for their base API	IoT network API: The level 2 APIs are derived from two general APIs at the level 1, and the level 2 APIs can be used substitutable for their base API at the level 1.
Interface Segregation	More client-specific interfaces are better than one general-purpose interface	IoT network API: At level 1, two APIs are created for IoT sensors and IoT actuators, given their different behavior.
Dependency Inversion	API should depend on abstraction but not on concretion	IoT network API: The implementation of IoT sensors and actuators is hidden at level 1. The manipulation of IoT devices for specified purposes at the level 3 reply on the level 2 API as interface. Any modification of lower-level communication would not impact the level 3.
Stateless	The server must not manage states between requests	Weather forecast API: Each request from the client thread contains all the necessary information for the server thread to process.
Cacheability	Requests that always return the same response must be cacheable	Weather forecast API: Each server response is stored in memory.

client thread and, therefore, is accessible to programming on the computing server.

3.4. API design principle and pattern in microservices

For the development of APIs in microservices, the software design pattern is an inevitable topic because a reliable and user-friendly architecture of the codes, including API and algorithms, improves the extensibility of the smart building design. Software design patterns Singleton and Proxy [39] impact the design of Weather forecast API. A server instance would be created as a single interface between all clients in the smart building and the NOAA public API service. This single instance would regularly retrieve the updated information from public API, renew the locally stored data, and respond to all requests in a queue. The server instance, the client instance, the NOAA public API, and local data storage in memory constitute a data system. The implementation of software development principles, including SOLID [40] and microservice API principles [34], is summarized in Table 4.

4. Applications for machine learning control research

Through the adoption of microservices and supporting ICT infrastructure, advanced control strategies can be tested safely without imposing risks on the building, enabling the on-site development and deployment of machine learning–based algorithms. Between 2019 and 2021, a series of laboratory studies examined controllable NV and TABS operation, illustrating how the microservice architecture allows the smart building to carry out data-driven investigations and experiments, including MPC, RL control, and coupled MPC approaches. Room air temperature and CO_2 concentration are measured using a wall-mounted sensor. The slab core temperature is monitored by an embedded temperature sensor installed within the concrete slab, while the slab surface temperature is measured by a portable temperature sensor.

4.1. Manual natural ventilation control with operable windows

The bottom manual windows were investigated with portable air sensors (Netatmo sensors in Table 2). The window opens and closes regularly to collect the room CO_2 and air temperature data. This initial study aims to understand the single-sided natural ventilation in the building and design a dynamic model to represent it. The results show that the consequence in short-term NV could be modeled by the autoregression system dynamics in CO_2 and room air temperature near the window area. A list of NV features is selected, and the identified NV system dynamics have a good performance in prediction. The natural ventilation dynamics are built in a control-oriented form, and data balance equations are constructed to reveal the quantitative relationship between the outside environment and the indoor environment. The details of this manual NV control experiment are available in [41].

4.2. Natural ventilation MPC

With the Tri-server architecture and IoT network API, the insight with manually operable windows is swiftly transformed into a second NV experiment and modeling. The CO_2 and temperature variation due to operation of motorized windows are identified through a data collection of 17 winter days as the training dataset [57]. A multi-objective optimization problem is formulated for natural ventilation MPC to co-optimize the room air quality and thermal comfort with the identified NV system dynamics. An on-site environment is organized with Tri-server architecture to avoid the distraction from the building operation signal in the rest of the building. An advanced math solver Gurobi with an academic license [61] is installed in the computing server and then integrated with IoT network API and CVXPY [62] for the predictive control programming.

Natural ventilation is regulated by a data-driven MPC scheme that uninterruptedly and periodically adjusts the opening position of two motorized upper windows to maintain indoor CO_2 concentration within prescribed limits during winter operation. The controller issues proportional window-opening commands (0–100 %) every 5 min based on linear prediction models of short-term CO_2 and air-temperature dynamics learned from on-site experimental data. MPC uses real-time inputs from rooftop weather sensors (outdoor temperature, wind speed and direction), indoor CO_2 and temperature measurements from a wall-mounted sensor, as well as occupancy sensor. By forecasting indoor conditions and enforcing constraints on air quality, temperature range, and allowable temperature drop, the MPC enables short-term single-sided natural ventilation events without causing unacceptable thermal discomfort or significant increases in heating energy use.

During the on-site tests in the lab, the occupancy sensor is involved in the dynamic CO_2 system. For example, on a typical winter day, the MPC regulates the room CO_2 level around 800–900 ppm while limiting the room air temperature variation due to introduction of cold fresh air (Fig. 8). The analysis shows that occupancy information can improve the predictive control performance by reducing window operation time. Another finding after 10 days test in winter is that short-term natural ventilation would not significantly increase the heating energy from TABS. Furthermore, the coupled system dynamics from the second experimental campaign are identified with a larger dataset, including that of first experimental campaign; the similarity between the two coupled system dynamics implies that the IoT network API enables a sort of online learning ability for the system identification procedure. More details of the Natural Ventilation MPC are available in [12].

4.3. Natural ventilation RL control

Experience from the successful real-building deployment of natural ventilation MPC also revealed a key limitation: multi-objective optimization relies on a mathematical solver that can respond to only one actuator at a time, creating a computational bottleneck that becomes significant in buildings with many actuators. To address this scalability issue, a RL framework was developed to replace the tested MPC structure. In the RL formulation, the CO_2 and room-temperature dynamics from MPC are embedded into reward functions, and the time-horizon optimization problem is reformulated as a statistical exploration process in probabilistic space, eliminating the need for a math solver. Two RL algorithms, Q-learning and deep Q-learning, are implemented through the IoT network API to learn optimal window-opening actions. The natural ventilation RL controller retains the same system configuration as the natural ventilation MPC and uses the identified thermal and

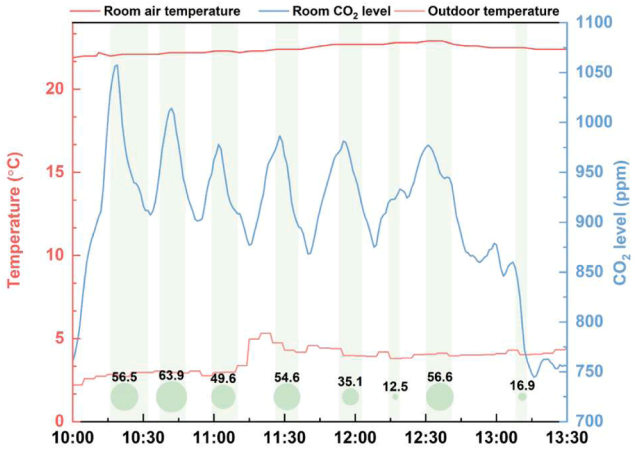


Fig. 8. Natural ventilation MPC onsite test with occupancy sensor. The mean outdoor air temperature during test is 3.7 °C. Shaded area represents the time period for south window opening. The bubbles represent the average opening angle during the shaded period.

CO_2 dynamics from the MPC model; it determines window operations using either discrete action steps (10 % increments) in Q-learning or continuous Q-value estimation in deep Q-learning.

Q learning has comparable control performance in the on-site test with predictive control. For example, Q learning controller regulated the room CO_2 level around 700–1000 ppm while avoiding cold draught on a typical march day in Boston area (Fig. 9). Due to the limitation of the computing server (Intel Xeon 2.40 G Hz with 4 G memory), the deep RL performed a non-neglectable delay in control performance. These results show a well-designed AI algorithm could perform at the same level as the optimization-based control with a math solver. More details of the natural ventilation RL control are available in [42].

4.4. Augmented TABS MPC

One augmented TABS control strategy is designed in the lab to enhance the energy efficiency of existing rule-based TABS control system, which uses only the embedded slab sensor with a weighted outdoor temperature in the next 48 h. As a high inertia system with a time constant of 10–15 h, the TABS system utilizes the concrete floor slab as the thermal mass and receives heating water supplied by a geothermal heat pump. The supply water temperature of the TABS is maintained at approximately 8–12 °C during winter operation and around 19 °C in summer. The total designed flow rate of the system is 2.55 m^3/h. The details of the rule-based TABS control system are available in [33]. However, the TABS overheating was experienced in the south-facing lab, when the solar radiation heats the room through the window.

The augmented control uses an MPC framework to override the existing rule-based control and prevent overheating caused by solar radiation. The actuator for the TABS system is the heating valve, which is governed by an on/off control signal. The primary control reference is the slab floor surface temperature, which is estimated as a virtual sensor based on embedded slab core temperature and room air temperature. During system identification, portable sensors, including a surface temperature sensor and an additional weather station, are used to collect slab surface temperature and solar radiation data. These data, capturing the slab's thermal behavior under direct solar exposure, are gathered as a 110-day training dataset. After the model is identified with the machine learning technique, the on-site experiment programming is realized with microservices IoT network API and weather forecast API. The weather forecast API calculates the future solar radiation with true sky conditions. The slab surface temperature dynamics could predict the future temperature with estimated solar radiation. The TABS could be open or closed in advance. Consequently, the involvement of solar radiation prediction in lab control could save TABS heating energy on sunny winter days. More details of the augmented TABS MPC are available in [43].

This newly developed TABS control is named Augmented TABS MPC because it can overwrite the running rule-based control, while the whole smart building control system is not disturbed. With Tri-server architecture and microservices, the augmented TABS control could override the valve in the lab by sending the control command more frequently (Appendix B). As shown in Fig. 10, one onsite test on a typical winter day demonstrated the augmented TABS control can predict the heating effect by solar radiation and plan to benefit it by closing TABS heating around 10 am, while avoiding overheating between 11 am and 2 pm. The slab surface temperature was impacted directly by solar radiation, but the slab core temperature increased relatively slowly. The room was, therefore, sufficiently heated without TABS. This augmented TABS control has considerable energy-saving potential for cold climate areas, as it uses a virtual slab surface temperature sensor as a control reference, which considers the built environment, such as building orientation, wind size, and natural environment, such as solar radiation.

4.5. MPC for Coupled natural ventilation and TABS

Building on the previously implemented standalone natural ventilation MPC (Section 4.2) and the augmented TABS MPC (Section 4.4), a coupled control strategy was developed to coordinate window-driven ventilation and radiant heating within a unified framework. In this scheme, the natural ventilation MPC shares its predicted room air temperature sequence with the augmented TABS MPC, allowing the TABS controller to incorporate ventilation-induced thermal effects into its slab temperature dynamics when determining valve operations. The coupled MPC retains the same system dynamics models as the standalone controllers; therefore, no additional model training is required. The integration is realized at the programming level without modifying the underlying models, enabling a rapid and seamless transition from standalone to coordinated control under the microservice architecture.

The coupled MPC is structured as two interconnected modules, each governing its respective actuator: the NV module generates proportional window-opening commands (0–100 %) at five-minute intervals, while the TABS module issues on/off heating-valve commands every 15 min. Through the exchange of predicted room air temperature, the fast-response ventilation control informs the slower radiant system, ensuring coordinated operation across different time scales. An additional supervisory module mitigates potential overheating during periods of high solar gains. Overall, the framework embodies a hierarchical control philosophy that harmonizes subsystem interactions to maintain indoor environmental quality and stable thermal performance.

The detail of coupling algorithm is presented in Appendix C. As

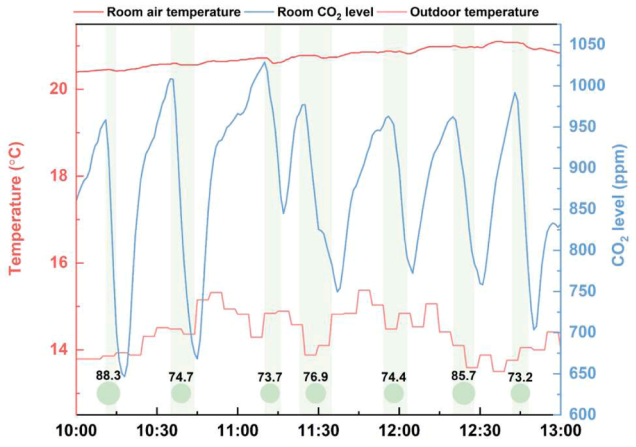


Fig. 9. Natural ventilation reinforcement learning control onsite test on Mar 24, 2021.

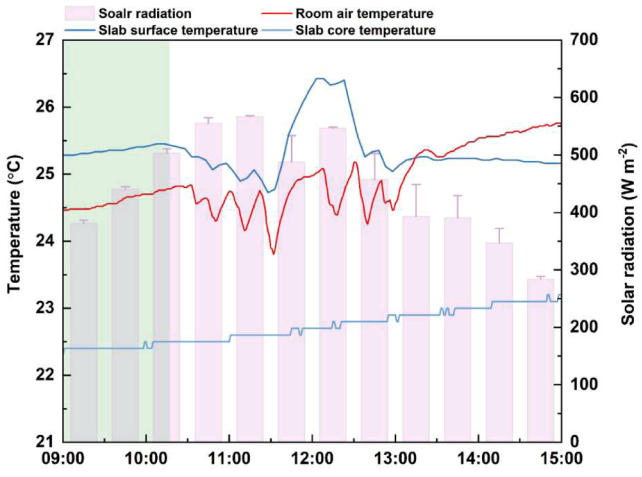


Fig. 10. Augmented TABS control. Green shaded area represents the TABS operation period. Purple bars represent the one-hour average of solar radiation.

shown in Fig. 11, one on-site evaluation was conducted on a typical winter day. The mean outdoor temperature from 9 a.m. to 4 pm. was 7.4 °C. The Lab was occupied before noon. The room CO_2 level was regulated by natural ventilation control. Meanwhile, the augmented TABS control regulated the heating valve by considering the solar radiation from the south window. As the room air temperature continued to increase after 12 pm, the south window was operated to maintain the room air temperature when the room was unoccupied.

4.6. Resolution

The primary objective of Section 4 is to demonstrate how microservices facilitate advanced research employing machine learning algorithms. A comprehensive on-site experimental evaluation of each control algorithm lies outside the scope of this paper. Instead, we provide several concise remarks to outline the key considerations.

In addition to controlling indoor CO_2, the natural ventilation MPC and RL strategies aim to prevent cold draughts from open windows. On-site tests (Fig. 8 and 9) show that maintaining CO_2 levels in winter had minimal impact on room air temperature. Ventilation effectively reduced occupant-generated CO_2, while the building's thermal mass (roof and wall) and solar gains helped stabilize indoor temperatures. For example, in Fig. 8, a window opening of 63.9 % after 10:30 corresponds to an angle of 11.5° out of a maximum 18°, slight temperature increases from slab heating underscore the importance of solar-induced thermal gains, which motivated the development of the augmented TABS MPC to account for these effects.

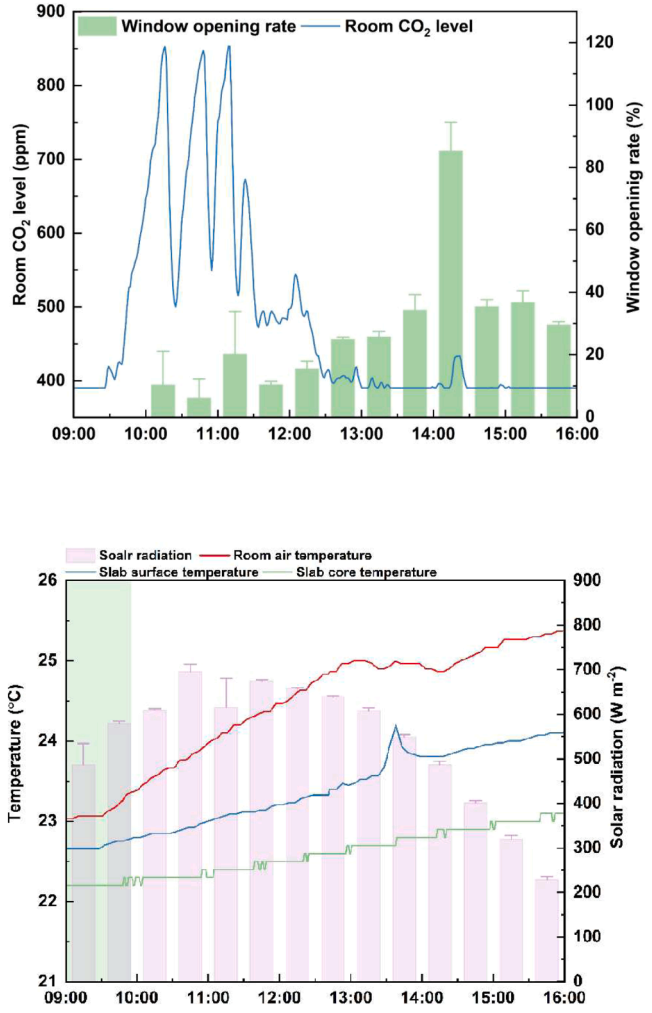


Fig. 11. Natural ventilation and TABS coupling control. Green shaded area represents the TABS operation period. Purple bars represent the one-hour average of solar radiation.

For natural ventilation MPC, evaluation across five tests shows that the controller can maintain room CO_2 level below the prescribed threshold, and no heating energy penalty was observed [12]. For natural ventilation RL control, evaluation across two tests indicates that its performance in regulating CO_2 levels and thermal comfort is comparable to that of natural ventilation MPC [42]. For augmented TABS control, an energy saving of approximately 200 to 300 W during winter daytime with solar radiation was identified, while simultaneously resolving the room overheating issue [43]. Because the augmented TABS MPC for the room operates together with the baseline rule-based TABS control for the building and improves it by overwriting the baseline control signal, the observed thermal comfort and energy savings of the augmented TABS control are well supported by the test results.

The coupled natural ventilation and TABS MPC is developed by integrating the natural ventilation MPC and the augmented TABS control using the same system dynamics, with the expectation that the fast response of the natural ventilation control algorithm will support the slow response of the TABS system and thereby improve the overall reaction speed of the coupled controller. On-site testing (Fig. 11) showed that the coupled system behaved similarly to the two standalone MPCs: indoor CO_2 concentration remained below 900 ppm, cold draughts were completely avoided, and the heating valve stayed closed in anticipation of strong solar radiation in the following hours. Compared with the rule-based TABS control, which kept the heating valve open until 12:17 pm according to the control script execution record, the coupled MPC achieved more than two hours of heating energy savings (at least 200 W based on routine heating demand) on the testing day. Further on-site experiments will be required to fully understand how the coupled MPC may outperform two independently operating natural ventilation MPC and augmented TABS control, and the microservices framework presented in this study has demonstrated its capability to support this type of control research and on-site experimentation.

5. Discussion

5.1. Why microservices architecture is important for smart buildings

The microservices architecture mirrors the structure of business units within a company, and this concept also applies to smart buildings. Typically, smart buildings incorporate a variety of information and communication devices. In competitive market economies like the U.S., no single tech company can supply all the devices needed for a smart building. Furthermore, for architects and researchers aiming to customize or enhance existing smart building solutions, they must find ways to modify closed systems and conduct more experiments to understand the buildings in specific environments better. This task is challenging for architects and researchers but understandable, as modifying smart building technologies often requires significant resources from the technology providers, such as industrial design and experimentation. Microservices architecture offers a solution by allowing each smart building technology to be reorganized as an independent microservice within the IoT network. In addition, essential public services like the National Weather Service have adopted API web services to disseminate critical information for smart buildings. Implementing API-based microservices architecture is, therefore, a necessary component of smart building technology.

Furthermore, microservices can function as a plug-in control mechanism. Microservices enable NV control experiments without requiring permanent modifications to the vendor-designed smart building architecture. When advanced control or machine-learning-based control strategies need to be deployed, the corresponding research scripts can be encapsulated within microservices and activated as needed, allowing the building control architecture to be temporarily reconfigured from a "dual-server" to a "tri-server" configuration for experimental purposes.

Upon completion of the experiments, the system can be reverted to its original vendor-defined operational mode. In this manner, the commercial agreements between researchers and vendors are respected to the greatest extent possible.

5.2. Comparative analysis of HVAC control case studies with respect to IoT architectures

Several ICT standards development organizations have published technical specifications forIoT systems, including the IEEE Standard for an Architectural Framework for the Internet of Things [18] and the ITU-T Recommendation Y.4000/2060 [44]. In parallel, a wide range of IoT architectural models have been proposed in the literature, such as the three-layer architecture (perception, network, and application layers) [45], the four-layer architecture (IoT devices, communication protocols, data processing, and IoT application layers)[46], the five-layer architecture (objects, object abstraction, service management, application, and business layers) [47], the six-component architecture (sensor, actuator, device, gateway, IoT integration middleware, and application)[48], and the seven-component architecture (sensors and actuators, IoT platform, communication network, IoT gateway, applications, management, and security)[45]. To further analyze how information flows among components or layers within IoT systems, Bahun et al. [46]. classified IoT architectures into three design topologies: (1) hub-based topology, in which all devices are centered around a local hub; (2) cloud-based topology, in which devices communicate directly with a cloud-based server; and (3) hybrid topology, which combines characteristics of the previous two approaches.

Recent developments in AI-driven control approaches integrated with IoT have been extensively reviewed in the literature [49]. While mechanically driven HVAC optimization using AI and IoT has emerged as a promising solution for improving energy efficiency in smart buildings [50], natural ventilation is still largely discussed in terms of its theoretical energy-saving potential, with comparatively limited practical implementation and validation [49].

In studies of advanced HVAC control, the underlying IoT and building energy management system (BMS) architectures are often insufficiently described, making it difficult to identify overall system designs or required modifications for control experiments. This limitation arises because on site experiments typically use only a subset of the building IoT infrastructure and because algorithm focused studies do not require full architectural disclosure to convey conceptual contributions, resulting in limited insight in both experimental and simulation-based work. This gap is evident in recent reviews of field demonstrations of MPC and RL for HVAC, where experimental studies accounted for only about 4 % of 2892 publications since 1997 and where attention focused on control architectures rather than IoT or BMS design [51]. Although control authority and system integration [52] were identified as major challenges for field deployment and commercialization, the supporting IoT and BMS architectures remain largely inaccessible and underexamined in the HVAC control literature.

In an on-site evaluation of MPC for mechanically driven HVAC systems [53], the MPC was implemented as an add-on controller that optimized cooling power and interfaced with local PID controllers for valve regulation. Although the study reported seamless switching between MPC and original thermostat control, detailed descriptions of the building IoT architecture were not provided. Nonetheless, the introduction of additional sensors and control modules implies modifications to the original IoT system, with the MPC effectively operating as a local hub. Within a three-layer IoT architecture, the added sensors and MPC controller can be interpreted as belonging to the perception and application layers, respectively.

In another real world implementation [54], the MPC algorithm was deployed on a cloud server and integrated with an existing BMS, with an IoT architecture comprising the BMS, a remote gateway, an IoT Core, rules engines, lambda functions, external APIs, and Z Wave nodes and sensors. The integration of MPC with the existing HVAC infrastructure was still considered a significant challenge. The cited research [54] shared common features with the present paper, such as system integration through a cloud server, but likely due to its focus on MPC experimentation, the IoT architecture was not described in detail or explicitly presented as a microservice based architecture design.

By contrast, in simulation-based studies [55], the IoT architecture is typically abstracted to a minimal set of sensors and actuators, and HVAC systems are modeled primarily through setpoint temperature control, providing limited insight into practical deployment within real smart building IoT infrastructures. Collectively, these studies highlight a common gap in the literature: while advanced control algorithms are well developed, their integration with and impact on building IoT architectures remain insufficiently characterized.

This paper provides valuable insights into how IoT architectures can support advanced control experiments for natural ventilation. The IoT architecture is explicitly articulated: the existing Dual-server architecture corresponds to a hub-based topology, whereas the proposed Tri-server architecture represents a cloud-based topology. Microservices serve as the enabling mechanism to transition between these two IoT network topologies, thereby facilitating the deployment of advanced natural ventilation control strategies within a smart building. The architectural principles and implementation details presented in this work can inform and support other researchers in the design and preparation of natural ventilation control experiments, as well as mechanically driven HVAC control experiments.

5.3. A generalized tri-server architecture for scalable and interoperable smart building IoT systems

The Tri-server architecture emerges from adapting the existing ICT framework to accommodate microservices within the traditional dual-server arrangement. This proposed architecture enhances the conventional Dual-server model, which is commonly found in modern smart building systems, by integrating microservices to improve flexibility and computational capability. Dual-server configurations typically reflect the multi-vendor nature of contemporary building technologies, where reliance on a single vendor is unrealistic. Protocols such as BACnet therefore play a crucial role in enabling interoperability across diverse systems, making the Dual-server architecture an effective representation of cross-vendor integration.

In more advanced implementations, the term “dual” reflects two functional server categories: a centralized communication and control server, and a set of distributed local servers that manage sensors and actuators associated with specific vendors. Introducing microservices transforms this arrangement into a Tri-server architecture by adding a third server dedicated to experimental control algorithms. This cloud-based platform provides AI-compatible computational resources that support advanced optimization engines and machine-learning-based control strategies. As a result, the system’s capabilities expand beyond those of the original Dual-server architecture, enabling uninterrupted control experimentation over an extended duration.

From a scalability standpoint, the IoT microservice layer simplifies the integration of heterogeneous sensors and actuators through the centralized communication hub. For example, devices from multiple vendors can be incorporated within the Tri-server framework for experimental control. Microservices offer a unified and flexible programming interface, allowing researchers to centrally manage all IoT devices across the network.

The transition from a dual-server to a tri-server configuration should be determined on a case-by-case basis, depending on the building’s operational and research requirements. Nevertheless, the design philosophy underlying the Tri-server model can be generalized for broader IoT network designs in smart buildings. In the generalized Tri-server architecture as shown in Fig. 12, the IoT network consists of three categories of servers. One server is designated as the centralized server

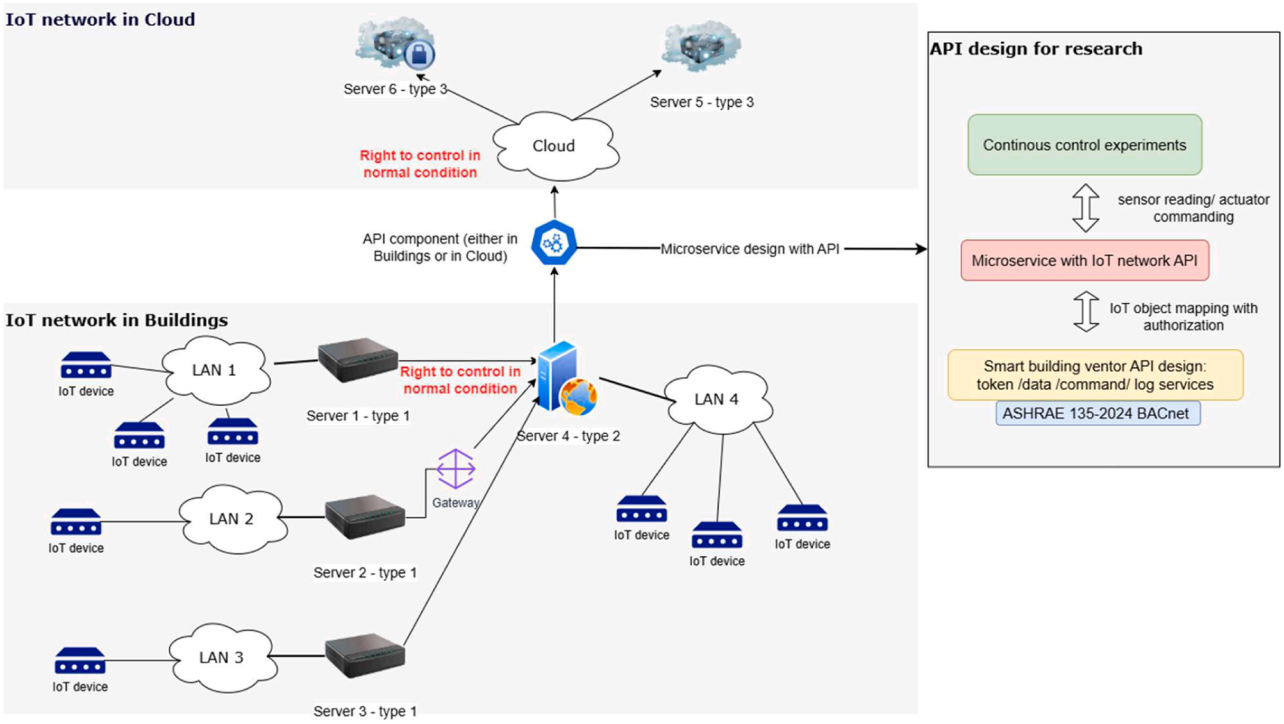


Fig. 12. Generalized Tri-server Architecture.

(type-2), equipped with vendor-specific APIs. All remaining servers in buildings are designated as type-1 servers, each responsible for managing IoT devices from different vendors and communication protocols. When a type-1 server uses a protocol incompatible with the type-2 server, a gateway is required, as illustrated by server 2 and server 4. Under normal operating conditions, the right to control or control authority is transferred from the type-1 servers to the type-2 server through appropriate configuration. However, device-protection logic and special pre-programmed control functions remain on each type-1 server to ensure safe operation during abnormal conditions.

Microservices extend the vendor API and expose a new API on the cloud server. Ultimately, under normal conditions, the right to control for the entire IoT network can be transferred to the cloud. One or multiple type-3 cloud servers may access the IoT network with appropriate security provisions. Importantly, the building's existing network topology can remain unchanged. Each type-1 server continues to interface with its respective IoT devices within dedicated LANs, and the topology of each LAN should be designed based on the spatial layout and HVAC functionality requirements of the building. Type-1, type-2, and type-3 denote three distinct categories of servers.

5.4. Holistic smart building design in three layers

Scholars have pursued the definition of smart building for decades, and this concept is evolving. In the 1990s, A smart building was a dynamic and responsive architecture, and it demanded intelligence applied at the concept, construction, and operation stages. Smart buildings can be technically as simple as vernacular architecture. For example, an igloo was a smart building for Inuit people because its shape and structure moderated the climatic impact [56]. In the 2010s, smart buildings integrated intelligence, enterprise, controls, and materials and construction as an entire, adaptable building system [57]. Recently, smart building concepts interact more with Information and Communication Technology (ICT) and AI technology in industry, such as "digitally connected structures" with "optimized building and operational automation"[58], "increasing digitalization of buildings" and "technology works seamlessly together to provide benefits for the people connected to the building"[59], "use data from IoT sensors to be responsive to occupant needs and optimize resources to increase comfort and productivity while reducing cost and waste"[60], and "help save energy by networking intelligent building technologies, increasing the comfort and convenience of users, and ensuring consistently reliable operation"[61]. It tends to be a self-fulfilling prophecy, and each definition underscores a specified focus in related research or business.

By adopting microservices architecture, this study proposes a comprehensive design approach for smart buildings across three levels: the building, ICT with microservices architecture, and intelligence. At the building level, the primary task is to define the project objectives, such as creating a high-quality, productive office space or an experimental research facility. It is essential to anticipate and plan for the return on investment and project costs. Sustainable architecture features dictate the scope of smart building behavior, often combining active and passive elements. For instance, TABS integrates a mechanical heat pump (active) and concrete thermal mass (passive) to provide various functions like free cooling, forced cooling, and heating. NV includes motorized operable windows (active) alongside manual operable windows, solar chimneys, and internal structural renovations (passive). Lighting design involves using glass for internal partitions, external shading, and basement lightwells (passive) with occupancy sensors, electrical lighting, and motorized skylight sunscreens (active). These architectural features aim to create a healthy, comfortable environment for occupants and cost-effective operations for building owners. AI-based research, including machine learning algorithms, has highlighted the energy-saving potential of NV and TABS. However, these features also add to the complexity of smart building design, especially during system integration. Manufacturers often protect their detailed designs and offer closed products with limited interfaces. Customizing engineered systems for individual buildings is resource-intensive for manufacturers, while architectural designs must meet diverse client requirements. Onsite installation engineers and behind-the-scenes design engineers follow different training and objectives, complicating the understanding of the installed system's full potential in a specific smart building.

At the ICT level, the sensor network and controllers are integrated

within the communication network, making all active architectural features like heat pumps, motorized windows, and motorized valves controllable and their statuses accessible. ICT in buildings evolves with social and technological advancements, including Shared Tenant Service, LAN, Fieldbus, Web 4.0, and IoT. More active building systems lead to more communication protocols for a given smart building. Building operators can access the server via the internet. Through the integration of KNX, Modbus, and BACnet, all active components and sensors are represented as BACnet objects. The IoT network API, designed as a microservice, converts these BACnet objects into Python functions and objects as IoT devices, enabling flexible programming for machine learning algorithm research. The Tri-server architecture integration emphasizes hardware and configuration, while developing the IoT network API focuses on programming, adhering to software design principles.

At the intelligence level, the decision-making process for operating a building involves both human and machine intelligence. In common building automation design, the system with Direct Digital Control (DDC) uses human intelligence to understand building comfort and energy efficiency, translating human operational experience into algorithmic programming. The Tri-server architecture handles building control under normal weather conditions and relies on DDC only for safety during extreme weather. All algorithmic programming for systems like NV and TABS is centralized in the computing server. Through the IoT network, the server collects data from the building, its environment, and its natural surroundings. A cloud computing paradigm-based weather forecast API provides crucial information to the server, aiding building operations. Researchers use machine learning algorithms and data science techniques to conduct AI-based research, transferring their insights into algorithms. Meanwhile, the computing server enhances human perception with IoT networks, accumulated big data, and online learning capabilities. Machine intelligence, or AI, develops as the smart building uncovers hidden patterns about itself and its environment under human guidance (Fig. 13).

Extreme weather would also be considered at the intelligence level as one option in the future, if the corresponding data is collected during the extreme weather and the extra requirements (such as energy saving) are expected for the extreme weather conditions. This alternative design would largely reduce the complexity of control systems by removing completely the DDC and the attached device-level control algorithms from the vendors. Consequently, the building designers would be responsible for the intelligence level design for all circumstances that a building encounters during its life cycle. In industrial building design, such as power plant buildings, high transparency in control systems and the study of all building working conditions are prerequisites for building designers (based on the lead author's professional experience), who are responsible for re-designing the control system architecture as system integration. Designers in commercial and residential buildings should learn from industrial building designers and customize the intelligence level design from a holistic perspective between buildings and the environment.

5.5. *Customized software architecture for smart buildings*

Every building has its unique environment. In each climate region, the integration of natural ventilation control in smart buildings requires a new customized software architecture, which is like architectural design in functional and aesthetic perspectives. Software engineering for smart buildings should evolve from HVAC-centered energy management system, which is only open for limited configuration along with electro-mechanical systems, to a building-centered climate-responsive learning system, which is flexible, modular, and extendable as the environment evolves. In natural ventilation control research, this customization can be satisfied by research software engineering (RSE), which focuses on the development of software for research purposes [62]. Meanwhile, the on-site experiments with RSE would enable programming to learn true knowledge from the real world and find the insights through the interaction with the real world.

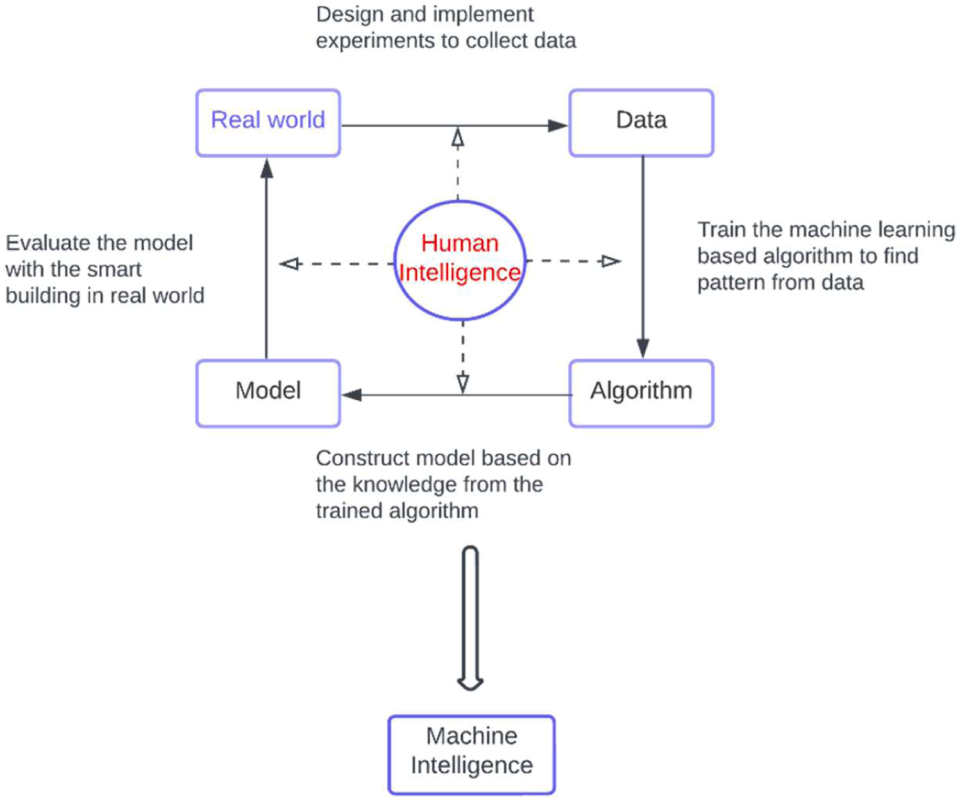


Fig. 13. Buildup of machine intelligence (or artificial intelligence) in buildings.

In a smart building, multiple software systems are possibly involved, such as ventilation control, heat and cooling control, water system control, lighting control, security, fire protection, solar generation, and grid-related control systems. Each control system, as software, is designed for a limited function and maintains essential communication with other software. In the ideal intelligence design for smart buildings, the software architecture needs to maximally reveal the potential of each building component and system, and coordinate with machine learning algorithms for optimal operations in all circumstances. The integration of software systems would be a roadblock to new smart building designs. In this context, if each software system is designed as a modular microservice, the building designers can organize a more flexible and effective software architecture with those microservices. Furthermore, with the rapid development of generative AI in the machine learning field, the software architecture is also responsible for AI Orchestration, which means managing various AI components to work together effectively within a larger system [63].

5.6. Future role of technology vendors in building design

"To create architecture is to put in order."[64] The intelligence level design will bring a new order in Architecture through smart buildings. From a building science research perspective, this paper demonstrates how commercial control systems in office buildings can be transformed into a control research platform, enabling data-driven experimentation. This adaptation demands a broader scope and greater responsibility than conventional commercial building design. It requires expertise in industrial control architecture, software engineering, scientific experimentation, and machine learning that are not yet fully integrated into the current design paradigm. Furthermore, to benefit more commercial buildings with intelligence-level design through smart building technology, the second challenge lies in translating this control research platform into commercial solutions for buildings. For most commercial building tenants and owners, the former may be comfortable at minimal cost, while the latter often prefer a simpler and possible cheaper control strategy to implement in buildings, shifting operation costs to the tenants. A cost-effective and low-barrier commercial solution is therefore valuable for both parties.

In response to both challenges, one direct approach is to expand the scope of building design education to encompass not only the classic architectural design but also control system architecture and intelligence-level design, including machine learning, generative AI algorithms, and AI system operations. With interdisciplinary insight across these domains, architects could offer holistic smart building solutions as part of their commercial practice.

However, integrating such a broad range of knowledge into architectural training may place an undue burden on both academic curricula and professional development. A more practical path at this stage is to delegate intelligence-related design components to technology vendors. These vendors must evolve beyond traditional HVAC-centric control systems toward data-driven technologies and scientific experimentation. Their responsibilities would include developing machine learning algorithms, conducting real-world experiments to discover novel building control strategies, and managing the trade-offs between scientific uncertainty and the maturity of deployable solutions throughout a smart building's lifecycle.

Collaboration between architects and technology vendors will be essential in crafting commercially viable solutions for building owners and tenants. Yet, the division of responsibilities within this partnership warrants further discussion. The boundary between these roles should not be seen as a boundary of knowledge, as the emerging paradigm of AI-integrated building design demands a deeply unified and interdisciplinary body of expertise. For architectural education and practice, the control system design might be the first step toward creating more effective AI-integrated buildings.

5.7. Contribution, implications and limitations

This study demonstrates how an existing smart building IoT architecture can be transformed to support advanced and AI-based control experiments for natural ventilation (NV), a capability that remains rare in operational buildings. Through coupled MPC control of NV and TABS, the study further verifies the practical feasibility of implementing coordinated predictive control across interacting building subsystems. This work advances a previously identified research direction [33], which called for embedding the logic of decoupled rule-based strategies into learning-based framework to achieve fully coupled NV–TABS operation for improved thermal comfort and energy performance. Under the conventional winter rule-based strategy, NV and TABS operate independently: NV is triggered by a CO_2 threshold (capped at 1200 ppm and adjusted according to forecasted outdoor temperature) using short pulse ventilation, while TABS is regulated via a slab temperature setpoint of approximately 24.5 °C, also weather-adjusted. No information exchange exists between the two subsystems, resulting in structurally decoupled control.

In contrast, the proposed framework establishes predictive coordination between NV and TABS. The NV MPC generates a sequence of room temperature predictions over an extended prediction horizon, which is transmitted to the TABS MPC to inform its optimization process. By incorporating this forecasted thermal trajectory into its control horizon, the TABS controller internalizes the anticipated thermal impact of ventilation. This information-level coupling shifts subsystem interaction from reactive coexistence to anticipatory coordination. The key contribution of this study lies in demonstrating that coupled MPCs for interacting passive and active building systems can be operationalized in practice, thereby advancing toward true system-level coordinated control in real-world applications.

The central implication of this study is that microservices architecture is critically important in transforming complex building systems into scalable, resilient IoT frameworks. By reorganizing each smart building technology as an independent microservice within the IoT network, the system supports the necessary software design for AI Orchestration: managing various AI components to work together effectively within a larger system. Furthermore, the architecture provides a practical control research platform by extending the existing building automation network. This adaptation, exemplified by the Tri-server architecture, expands system capabilities beyond the conventional Dual-server model, enabling uninterrupted control experimentation over an extended duration. This framework allows researchers to conduct advanced control experiments on-site without imposing risks on the entire building. Specifically, the IoT network API microservice converts BACnet objects into user-friendly Python functions and objects, enabling flexible programming for machine learning algorithm research. Ultimately, the ability to automate and control complex systems like natural ventilation at the intelligence level is expected to encourage more eco-friendly passive design in future building projects.

Despite its architectural strengths, several limitations hinder the wider adoption and commercial applicability of the microservices platform for control experiments. The conversion of existing systems is constrained by industry norms in which vendors protect proprietary designs and offer closed products with limited interfaces, while commercial APIs restrict device-level access for safety and security reasons. As a result, adapting a commercial building system into a control research platform demands a combination of expertise in industrial control architecture, software engineering, and machine learning that is not yet commonplace in current practice. Moreover, translating such a sophisticated platform into market-ready solutions poses a significant challenge, as most building owners and tenants prioritize simpler and more economical control strategies. In essence, shifting from closed, vendor-driven systems to open, customizable microservices is akin to converting a tightly controlled factory into a flexible research laboratory: the opportunity for innovation is substantial, but it requires

overcoming proprietary constraints and integrating new technical skill sets into the traditional building management ecosystem.

6. Conclusion

A microservices architecture is proposed and implemented through the Tri-server architecture in a test building. The IoT network API microservice integrates sensors and actuators into advanced programming and cloud computing environments. The weather forecast API microservice can capture the short-term weather changes to inform the machine learning algorithm to guide the building to adapt to the changing ambient environment. The microservices enable the use of AI-based algorithms, including machine learning algorithms for automatic and smart control of heating, ventilation, and air conditioning systems. Data-driven experiments identify control-oriented dynamics of the natural ventilation system. Predictive and reinforcement learning controls are tested in the test building to optimize the indoor air quality and the thermal comfort. Solar radiation prediction is incorporated into the modeling of the thermally activated building system (TABS). Onsite tests demonstrate that the new TABS controls energy-saving potential. For the coupled TABS and natural ventilation system, the microservices architecture enhances the ability to explore the environment and uncover hidden patterns. The coming trends of data-driven algorithm implementation and generative AI techniques require effective software architecture, such as microservices, that can orchestrate all the building actuators while considering sensor data, public API, and intelligence from both human and machine.

The microservices can benefit the stakeholders of building projects, such as building owners and building control companies. In parallel, emerging technology vendors will take on responsibility for overseeing the associated design components and operational activities. Microservices, as a loosely coupled connection between ICT and intelligence levels, can guide the system integration in smart building projects. Taking natural ventilation control with IoT network API microservice in this study as an example, integrating control algorithms and on-site testing will be easier. During the on-site testing, the collected data can also be used to calibrate control algorithms through online machine learning techniques. The building control project can be scaled up quickly in the real built environment from a few test buildings to wider implementation. With the microservices, intelligence can better understand the built environment and better command the actuators at the ICT level. The ultimate optimal building operation can reveal the building's potential at the physical level, including the passive design. Progressively, the ultimate operation will encourage more eco-friendly passive design, such as natural ventilation. If natural ventilation becomes controllable automatically at the intelligence level, more building projects will implement natural ventilation effectively in the real world.

CRediT authorship contribution statement

Wei Zhang: Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Leslie Norford:** Writing – review & editing, Validation, Supervision, Project administration, Methodology, Conceptualization. **Wentao Wu:** Writing – review & editing, Visualization, Validation, Software, Project administration, Methodology, Investigation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Dr. Zhang would like to express sincere appreciation to Harvard Center for Green Buildings and Cities, and to Yihan Chen of the Massey University and MingTong Li of the University of Canterbury for assistance with figure preparation.

Dr. Wu would like to acknowledge the support of the Royal Society New Zealand through the Catalyst Leader grant with Program Contract Number: ILF-MAU2301. The grant enables his continuous support and research on the low-temperature heating system – thermally activated building system.

Appendix A: Complementary Detail of Microservices

Details of IoT network API functions

Table A4
Partial function list for IoT network API.

Lays	Function	Description
0	_gatewayIO	find the gateway IO for a specified IoT device.
	_getDevice	find the network location of a specified IoT device.
	_getValues	return values from a specified network location.
	_getCommand	prepare the command for a specified IoT device
	_getMetaData	return the property of a specified IoT device.
	_setValue	set a value at a specified network location.
1	tokenAPI	fetch a token with authorization information and diagnoses the connection with the API server if it fails to get a token. For system security, one token will expire after 10 mins of inactivity.
	sensorAPI	return the current measurement from a specified sensor, with token and sensor information as input.
	actuatorAPI	send the command to a specified actuator with token and actuator information as input. It would return True if the actuator is in good status.
2	controlAPI	send a command to a specified actuator, and gets the status from the attached sensor, by using the functions: sensorAPI and actuatorAPI
	outdoorT	wrap the outdoor temperature sensor information with sensorAPI. It works with a token as input and returns the current measurement of outdoor temperature.
	Windspeed, winddirection, etc.	As outdoorT, all the IoT sensors are coded as an API.
3	crossNVAPI	send commands to two specified actuators for cross-ventilation. It uses parallel processing to ensure the actuators perform simultaneously.
	Isolation3_summer	close the windows and opens skylight shading simultaneously. It is designed to end the natural ventilation experiments in summer.
	Isolation	close the windows and valve and opens skylight shading simultaneously. It is designed to end natural ventilation and TABS.

(*continued on next page*)

Table A4 (*continued*)

Lays	Function	Description
	_getTrendSeriesInfo	fetch all the historical trends for a specified IoT sensor.
	_getTrendSeries	fetch the historical trends within a given period for a specified IoT sensor.
	Occupied_verify	use room window positions and CO_2 trends to check if the room is occupied.

The Python source code of the IoT network API is in Appendix B.1 of [34].

Estimation of Solar Radiation with Cloud Condition in Microservice 2 Weather Forecast API

The estimation of real solar radiation is realized in several steps: 1) calculate the fraction year (γ), equation of time (*eqtime*) and solar declination angle (*decl*). 2) calculate the solar time (*tst*), solar zenith angle (φ), and solar azimuth (θ). 3) calculate the air mass (AM) from the solar zenith angle. 4) calculate the clear sky irradiation (G). 5) estimate the solar radiation with the clouds (G_{cloud}).

$$\gamma = \frac{2\pi}{365} * \left(day - 1 + \frac{hr-12}{24} \right) \quad \text{(A1)}$$

$$eqtime = 229.18 * (0.000075 + 0.001868\cos(\gamma) - 0.032077\sin(\gamma) - 0.014615\cos(2\gamma) - 0.040849\sin(2\gamma)) \quad \text{(A2)}$$

$$decl = 0.006918 - 0.399912\cos(\gamma) + 0.070257\sin(\gamma) - 0.006758\cos(2\gamma) + 0.000907\sin(2\gamma) - 0.002697\cos(3\gamma) + 0.00148\sin(3\gamma) \quad \text{(A3)}$$

$$tst = hr * 60 + mn + \frac{sc}{60} + eqtime + 4 * longitude - 60 * timezone \quad \text{(A4)}$$

$$\cos(\varphi) = \sin(lat)\sin(decl) + cos(lat)\cos(decl)\cos\left(\frac{tst}{4} - 180\right) \quad \text{(A5)}$$

$$cos(180 - \theta) = - \frac{\sin(lat)\cos(\phi) - \sin(decl)}{\cos(lat)\sin(\phi)} \quad \text{(A6)}$$

$$AM = \frac{1}{\cos(\varphi) + 0.00176759\ \varphi\ (94.37515 - \varphi)^{-1.21563}} \quad \text{(A7)}$$

$$G = \left(5.09 * 10^{-5} * al + 0.868\right) * I_0 * \sin(h) * e^{-\left(3.92*10^{-5}*al + 0.0387\right) AM\left(e^{-al/8000} + (TL-1)*e^{-al/1250}\right)} \quad \text{(A8)}$$

$$G_{cloud} = G\left(p + qC + rC^2 + sC^m\right) \quad \text{(A9)}$$

where, hr is the hour in 24-hour format (0–23), mn is the minute (0–59), and sc is the second (0–59); the calculation of fraction year (γ) in Equation A1 should use 366 in the denominator; *longitude* is the site's longitude;timezone is in hours from UTC; *lat* is the site's latitude; the calculation of fraction year (γ), equation of time (*eqtime*) in Eq. (A2), solar declination angle (*decl*) in Eq. (A3), solar zenith angle (φ) in Equation 5, and solar azimuth (θ) in Eq. (A6) based on low accuracy equations from [65,66]. In Eq. (A7), AM is the optical path length through the atmosphere, it is calculated in Gueyemard's model [67,68]. In Eq. (A8), G is the clear sky solar irradiation which is calculated with Ineichen's model [69], *TL* is the Linke turbidity coefficient [70], *al* is the site's altitude, I_0 is the normal incidence extraterrestrial irradiance, h is the solar altitude which can be derived from solar zenith angle (φ). In Eq. (A9), G_{cloud} is the solar irradiation with clouds, which is calculated with a regression model with coefficients [71], and *C* is the cloud amount in okta (0–8). The cloud amount information from NOAA is described as "mostly sunny," "partly cloudy," "partly sunny," "mostly cloudy," "considerable cloudiness," etc. The quantification of sky conditions is based on the explanation from NOAA [72]. A complete realization in Python of this solar radiation estimation with clouds, and the weather forecast API is in Appendix B.2 of [34].

Usage of Microservice 1 in Control Script Programming

```
token = tokenAPI()
print("outdoor temperature is:",outdoorT(token))
print("room temperature is:",z31roomT(token))
controlAPI(token, south_window, 20)
```

```
Token received.
outdoor temperature is: 25.9400005340576
room temperature is: 24.3400001525879
South Window
Command is sucessfully sent to actuator:  20
South window position
The actuator South Window is working and its position is: 0.0
South window position
The actuator South Window is working and its position is: 7.0
South window position
The actuator South Window is working and its position is: 16.0
```

```
token = tokenAPI()
t0 = time.time()
command = 0
Hours = 10

while (time.time() < t0 + Hours*3600):

    CO2_value = z31CO2(token)
    print("CO2 level is",CO2_value)

    if CO2_value > 1000:
        position_to_set = 100
    elif CO2_value < 800:
        position_to_set = 0
    else:
        position_to_set = command

    command = position_to_set
    controlAPI(token, south_window, position_to_set)
    time.sleep(300)
```

Fig. 14. two API examples. Left: read sensors and send command; right: CO_2 demand control with token management strategy.

Two programming examples, the first is to read outdoor temperature and room temperature sensors, and open south window with a fixed value. The second is a simple CO_2 demand control, which opens the operable window based to room CO_2 level.

This document includes two programming examples: one that retrieves outdoor and room temperature data and activates the south window at a predefined threshold, and another that performs simple CO_2 demand control by opening the operable window when indoor CO_2 exceed a specified level.

Appendix B: ICT level adaptation: Tri-server architecture

This appendix details the Information and Communication Technology (ICT) level adaptation and the transformation of the smart building's operational framework for research purposes in the Lab room located on the 3rd floor. This integration and adaptation work occurred between 2019 and 2021.

Dual-server Architecture for Smart Building Operation

Two major vendors that participate in the smart building operation integration are anonymized as A and B. Vendor A delivered a natural ventilation system equipped with sensors, actuators, and a dedicated management server (Server A). Vendor B supplied a building management system with its own server (Server B) and seamlessly integrated all building subsystems—Server A included—into its centralized software platform.

The integrated smart building operation system, referred to as a Dual-server architecture, comprises two servers: Server A and Server B. Vendor A and Vendor B collaborated to integrate the natural ventilation system (Server A, based on KNX) with the building management system (Server B, based on BACnet). A cloud-based virtual machine (VM), hosted by a leading cloud service provider, provides remote access to both servers. The API on Server B is exposed to the VM, secured by VPN access controls within the campus network. All sensors and actuators from Vendor A are successfully represented within Vendor B's management interface. This seamless integration ensures that the Dual-server architecture supports comprehensive smart building monitoring.

Despite its functionality, the Dual-server architecture presents limitations for control-oriented research. During standard operations, the system executes pre-programmed, rule-based algorithms configured independently by Vendors A and B. In this setup, any control command issued from Server B to Vendor A's actuators is promptly overridden by Server A, which retains command precedence whenever both servers have communication access to the same devices. Moreover, Server A lacks API support, restricting administrative control to manual operations through its graphical user interface. These constraints hinder the flexibility and experimental control required in advanced smart building research.

Furthermore, the API service provided by Server B is secured through token-based authentication. Once the token expires, any control commands issued from external scripts are automatically rejected. This limitation, combined with the lack of API support from Server A and the rigid rule-based control hierarchy, renders the Dual-server architecture unsuitable for control-oriented research. Notably, neither Server A nor Server B was originally designed with experimental control functionality in mind.

Tri-server Architecture for Research

In this three-floor smart building, the Lab was originally designed for control research. Given the limitations of the Dual-server architecture, one viable pathway for enabling control research involves leveraging Server B and its API to interface with the Lab's actuators: window motorline, skylight motorline, sunscreen motorline, and TABS valve.

As part of the lead author's PhD work, a framework for control experiments was independently developed and deployed within the Lab, forming what is now referred to as a Tri-server architecture. This framework operates autonomously and does not require system-level collaboration from either Vendor A or B. The transition from Dual-server to Tri-server relies primarily on reconfiguration and the integration of an IoT-based network microservice for API orchestration.

The three motorline types—window, skylight, and sunscreen—are centrally managed as part of the building's natural ventilation function. For these motorlines, the strategy is to manually isolate rule-based control commands from Server A and repurpose another communication channel to bind the motorlines directly to Server B. This ensures that Server A's control signals are fully excluded from the motorline pathways.

The TABS valve is handled differently in Server A than the motorlines. The valve relies on a temperature-sensing material heated electrically, resulting in a significantly slower thermal response compared to the electric motors driving the window actuators. To counteract Server A's intermittent impulse signals, Server B transmits a relatively high-frequency signal, forming a quasi-continuous control input that effectively neutralizes the rule-based impulses and stabilizes valve actuation.

Two distinct strategies have been developed for token management. The straightforward token strategy involves requesting a token prior to each command execution. This approach is well-suited for short bursts of intensive operations, such as actuator status checks, control mode configurations, and actuator testing. However, Server B imposes a quota limit on token issuance; when exhausted, subsequent token requests are temporarily rejected.

To address this constraint, the minimal token strategy aims to maximize token utility by extending its lifespan. Here, the token is periodically reactivated at a fixed frequency before its expiration, allowing the control program to maintain uninterrupted access while reducing the total number of token requests.

Through systematic observation and testing, these insights were consolidated to realize the Tri-server architecture. The resulting methods have been documented as a configuration and programming tutorial for the research center.

In the new architecture:

- Hardware management server: Server A links actuators and sensors, serving as the device integration layer.
- API service management server: Server B exposes a RESTful API for actuator and sensor access.
- Computing server: A cloud-based VM hosts the programming environment and optimization solvers, providing the control and optimization runtime.

Co-existence of Two Architectures in the Same Smart Building

While the Lab operates independently with a data-driven Tri-server architecture, the remainder of the smart building continues to function under the original rule-based Dual-server framework. This architectural coexistence introduces operational complexity, particularly during the Tri-server's development phase. The lead author concurrently served as both a smart building operator and control researcher, actively maintaining the separation

between the two architectures through targeted reconfiguration and customized programming.

During extreme weather events, Server A, part of the Dual-server architecture, assumes control with the highest-priority safety override signal, reasserting authority over all window motor lines in the Lab to safeguard building integrity. On another occasion, the system-wide modification or updates by Vendor A prompted the manual switches from Tri-server to Dual-server architecture, enabling engineering verification.

Appendix C: Natural ventilation and augmented TABS coupled MPCs

This appendix outlines the coupling algorithm used for the integrated natural ventilation and augmented TABS MPC control. The combined strategy is constructed from two standalone MPCs: one governing natural ventilation and the other governing TABS, working together to regulate indoor CO_2 concentration and thermal comfort through window operation and radiant heating. The system dynamics and modeling approaches follow directly from the formulations of the individual MPCs, as detailed in references [12,43]. In the coupled algorithm framework as below, the room-temperature trajectory predicted by the natural ventilation MPC (module 1) is passed to the augmented TABS MPC (module 2), and an additional supervisory layer (module 3) is incorporated to mitigate potential overheating using the same predictive input.

ALGORITHM: Natural Ventilation and TABS coupling control.

1 MODULE 1: WINTER NATURAL VENTILATION CONTROL (INTERVAL 5 MINS)
  Input: *room air temperature, room CO_2, wind direction & speed, outside temperature, occupancy sensor (optional)*
  Calculate the possible room air & CO_2 with possible window opening in a time horizon
  Select the optimal window opening with the optimization solver
  Output: *window opening U1, room air temperature upper bound sequence r, room air temperature sequence T*
2 MODULE 2: ROOM OVERHEATING PREDICTIVE PROTECTION (INTERVAL 5 MINS)
  Input: *room air temperature upper bound sequence r, room air temperature sequence T*
  Calculate the window opening 2 to avoid overheating in the room
  Compare window openings 1 and 2, select the max opening
  Output: *final window opening*
3 MODULE 3: TABS UNOCCUPIED CONTROL FOR SOUTH-FACING OFFICE (INTERVAL 15 MINS)
  Input: *predicted room air temperature sequence T (module 1), estimated solar radiation*
  Calculate the slab surface temperature with possible heating valve position
  Select the optimal valve position with optimization solver
  Output: *final valve position*
4 SEND THE FINAL WINDOW OPENING (EVERY 5 MINS) AND FINAL VALVE POSITION (EVERY 15 MINS) TO ROOM CONTROLLERS
5 **END**

The natural ventilation and TABS coupling control is constituted of three modules, including two MPC modules and one proportional control module. The source code of control experiment is available: https://github.com/zwei2016/NV_TABS_experiment

Module 1: Winter Natural Ventilation MPC

Objective function:

$$\min c_1 * \sum_{1}^{T_1} U_1(k)^2 + c_2 * \sum_{1}^{T_1} (U_1(k) - U_1(k-1))^2 + c_3 * \sum_{1}^{T_1} \alpha_1(k) + c_4 * \sum_{1}^{T_1} \alpha_2(k) + c_5 * \sum_{1}^{T_1} r_1(k) + c_6 * \sum_{1}^{T_1} r_2(k) \quad \text{(C1)}$$

With system dynamics of room CO_2 level and air temperature.

$$C_{roomair}(k+1) = a_1 * C_{roomair}(k) + U_1(k) * g_C + nG \quad \text{(C2)}$$

$$T_{roomair}(k+1) = a_2 * T_{roomair}(k) + U_1(k) * g_T \quad \text{(C3)}$$

Subject to:

$k \in [1, T_1]$

$800 - \alpha_1(k) \le C_{roomair}(k) \le 1000 + \alpha_2(k)$

$20 - r_1(k) \le T_{roomair}(k) \le 24 + r_2(k)$

$|T_{roomair}(k+1) - T_{roomair}(k)| \le 1$

$0 \le U_1(k) \le 100\%$

$c_{1\sim6}$: the coefficients

g_T: room air temperature system dynamic- linear function of natural ventilation features

g_C: room CO_2 system dynamic - linear function of natural ventilation features

G: CO_2 generation per person n: number of occupants in office

$T_{roomair}(k)$: the room air temperature, °C

$C_{roomair}(k)$: the room air CO_2 level, ppm

The MPC iterates to maintain the zone temperature. The MPC, including the optimization, is realized through Gurobi optimizer [64]. The prediction horizon is 5 min that is adequate for short-term airing with winter natural ventilation.

Module 2: Room air overheating proportional control

$$U_3(k) = k * r_2(k) * (T_{roomair}(k) - T_{outdoorair}) \tag{C4}$$

$T_{roomair}(k)$: the room air temperature, °C

$T_{outdoorair}(k)$: the outdoor air temperature, °C

$r_2(k)$: the soft upper bound of room air temperature from Module 1

Module 3: TABS augmented MPC:

Objective function

$$\min c_7 \sum_{1}^{T_2} U_2(k)^2 + c_8 * \sum_{1}^{T_2} (U_2(k) - U_2(k-1))^2 + c_9 * \sum_{1}^{T_2} \alpha_3(k)^2 \tag{C5}$$

with estimated room slab surface temperature dynamic

$$T_{surface}(k) = a * T_{surface}(k-1) + b * E_{TABS} * U(k) + c * E_{solar}(k) + d * T_{roomair}(k) + e \tag{C6}$$

$$T_{surface}(k-1) = G * T_{roomair}(k-1) + H * T_{slabcore}(k-1) - W \tag{C7}$$

Subject to:

$$k \in [1, T_2]$$

$$T_{surface}(k) \leq T_{limit} + \alpha_3(k)$$

$$\alpha_3(k) \geq 0$$

$$U_2(k) = 0,\ 100\%$$

T_{limit}: the high-temperature limit for slab surface, for example, 22.5 °C

U(k): binary decision variable for TABS heating valve, it should be either 0 or 1

$T_{surface}(k)$: the slab surface temperature, estimated as a virtual sensor based on $T_{slabcore}(k)$ and $T_{roomair}(k)$, °C

$T_{slabcore}(k)$: the slab concrete core temperature, °C

$T_{roomair}(k)$: the predicted room air temperature from module 1, °C

$E_{TABS}(k)$: TABS heating energy rate, W

$E_{solar}(k)$: solar radiation prediction in 15 mins, W

a,b,c,d,e, G,H,W: coefficients

The objective function (Eq. (C5)) is set up to minimize the TABS energy usage by TABS heating valve opening U(k) while limiting the slab surface temperature through $\alpha(k)$. U(k) - U(k-1) represents the change of valve opening during the control interval. The coefficient of $\alpha_3(k)$ is tuned as 200 to highlight the constraint of slab surface temperature, compared to the valve opening and the change of valve opening. The MPC, including the optimization, is realized through Gurobi optimizer [64]. The control time interval is 15 mins. The weather forecast API provides solar radiation prediction $E_{solar}(k)$ in 15 mins. More information of experiment setting is available [43].

Data availability

Data will be made available on request.

References

[1] H.P. Das, et al., Machine learning for smart and energy-efficient buildings, Environ. Data Sci. (3) (2024) e1.

[2] Z. Liu, et al., Advanced controls on energy reliability, flexibility and occupant-centric control for smart and energy-efficient buildings, Energy Build. 297 (2023) 113436.

[3] D. Rodríguez-Gracia, et al., Review of artificial intelligence techniques in green/smart buildings, Sustainable Comput. 38 (2023) 100861.

[4] W. Li, et al., Natural ventilation cooling effectiveness classification for building design addressing climate characteristics, Sci. Rep. 14 (1) (2024) 16168.

[5] D. Mortari, et al., Smart ventilation in residential buildings: a systematic review of control strategies and their effectiveness, J. Build. Eng. 108 (2025) 112584.

[6] B. Belmans, et al., Set-up and evaluation of a virtual test bed for simulating and comparing single- and mixed-mode ventilation strategies, Build. Environ. 151 (2019) 97–111.

[7] D.A. Bert Belmans, S. Verbeke, in: Amaryllis Audenaert, Filip Descamps, *A case study on residential mixed-mode ventilation using the Ventilation Controls Virtual Test Bed*, in *40th AIVC - 8th TightVent - 6th venticool Conference*, Ghent, Belgium, 2019.

[8] K. Grygierek, J. Ferdyn-Grygierek, Design of ventilation systems in a single-Family house in terms of heating demand and indoor environment quality, Energies 15 (2022) 8456, https://doi.org/10.3390/en15228456.

[9] C. Rizo-Maestre, et al., Intelligent ventilation and indoor air quality: state of the art review (2017–2025), Buildings 16 (2026) 65, https://doi.org/10.3390/buildings16010065.

[10] Y. An, Z. Niu, C. Chen, Smart control of window and air cleaner for mitigating indoor PM2.5 with reduced energy consumption based on deep reinforcement learning, Build. Environ. 224 (2022) 109583.

[11] E.X. Chen, et al., Adaptive model predictive control with ensembled multi-time scale deep-learning models for smart control of natural ventilation, Build. Environ. 242 (2023) 110519.

[12] W. Zhang, et al., Model predictive control of short-term winter natural ventilation in a smart building using machine learning algorithms, J. Build. Eng. 73 (2023) 106602.

[13] A.I. Khdair, et al., Advancing natural ventilation in sustainable architecture: mechanisms, innovations, and climate-responsive design for energy-efficient buildings, Renew. Sustain. Energy Rev. 226 (2026) 116314.

[14] Y. Chen, et al., An hour-ahead predictive control strategy for maximizing natural ventilation in passive buildings based on weather forecasting, Appl. Energy 333 (2023) 120613.

[15] H.L. Gough, et al., Evaluating single-sided natural ventilation models against full-scale idealised measurements: impact of wind direction and turbulence, Build Environ. 170 (2020) 106556.

[16] D. Etheridge, A perspective on fifty years of natural ventilation research, Build Environ. 91 (2015) 51–60.

[17] T. Erhart, et al., Experimental validation of basic natural ventilation air flow calculations for different flow path and window configurations, Energy Procedia 78 (2015) 2838–2843.

[18] *IEEE Standard for an Architectural Framework for the Internet of Things (IoT)*. IEEE Std 2413-2019, 2020: p. 1–269.

[19] ASHRAE, *Standard 135-2024-BACnet-A data communication protocol for building automation and control networks*. 2024.
[20] Siemens, *Building automation APIs*. 2025.
[21] V. Graveto, T. Cruz, P. Simões, Security of Building Automation and Control Systems: survey and future research directions, Comput. Secur. 112 (2022) 102527.
[22] A. Das, S. Chakraborty, S. Chakraborty, Where do all my smart home data go? Context-aware data generation and forwarding for edge-based microservices over shared IoT infrastructure, Future Gener. Comput. Syst. 134 (2022) 204–218.
[23] J. Gough, Mastering API architecture: design, operate, and Evolve API-based Systems, O'Reilly, 2022.
[24] M. Richards, Software Architecture Patterns, 2nd Edition, O'Reilly Media, Inc, 2022.
[25] S. Newman, BuildingMicroservices, 2nd Edition, O'Reilly Media, Inc, 2021.
[26] Community, G.P., *Microservices architecture: have engineering organizations found success?* 2023.
[27] C. Lira, et al., Architecture for IoT applications based on reactive microservices: a performance evaluation, Future Gener. Comput. Syst. 145 (2023) 223–238.
[28] D.S. Aamir Shakir, M. Volk, N. Jamous, K. Turowski, Towards a concept for building a big data architecture with microservices, in: 24th International Conference on Business Information Systems, 2021.
[29] Chamari, L., Pauwels, P., Petrova, E., Dubbeldam, J.W., de Jong, N., & Gunderi, K., *Reference architecture for Smart buildings*. 2023.
[30] L. Chamari, E. Petrova, P. Pauwels, An end-to-end implementation of a service-oriented architecture for data-driven smart buildings, IEEE Access 11 (2023) 117261–117281.
[31] L. Roda-Sanchez, et al., Cloud–edge microservices architecture and service orchestration: an integral solution for a real-world deployment experience, Internet of Things 22 (2023) 100777.
[32] W. Wu, J.M. Han, A. Malkawi, Simplified direct forcing approach for dynamic modeling of building natural ventilation, Build Environ. 188 (2021) 107509.
[33] B. Yan, et al., Comprehensive assessment of operational performance of coupled natural ventilation and thermally active building system via an extensive sensor network, Energy Build. 260 (2022) 111921.
[34] W. Zhang, Data-driven Predictive Control Optimization for Natural Ventilation in Buildings, PhD Dissertation, Harvard University, 2021.
[35] J. Grinham, et al., Zero-carbon balance: the case of HouseZero, Build Environ. 207 (2022) 108511.
[36] A. Malkawi, et al., Design and applications of an IoT architecture for data-driven smart building operations and experimentation, Energy Build. 295 (2023) 113291.
[37] H. Samuelson, C. Reinhart, Modeling an existing building in DesignBuilder/EnergyPlus: custom versus default inputs, in: Eleventh International IBPSA Conference, Glasgow, Scotland, 2009.
[38] W. Zhang, et al., Applying machine learning for building natural ventilation control, in: The 16th Conference of the International Society of Indoor Air Quality & Climate, 2020.
[39] E.F.E. Robson, Head First Design Patterns, 2nd Edition, O'Reilly Media, Inc, 2021.
[40] Martin, R.C. *Design principles and Design patterns*. 2000.
[41] W. Zhang, W. Wu, A. Malkawi, *Using machine-learning algorithms to improve CO2-based demand controlled natural ventilation*. Prometheus, in: **04**(4th International Graduate Student Symposium: Buildings, Cities, and Performance, II), 2020, pp. 34–41.
[42] Z. Wei, M. Ali, Simulation-based control of natural ventilation with operable windows: transformation from predictive control into reinforcement learning control, in: 2022 Building Performance Analysis Conference and SimBuild co-organized by ASHRAE and IBPSA-USA, ASHRAE/IBPSA-USA, 2022, pp. 113–125.
[43] W.W. Wei Zhang, L. Norford, A. Malkawi, A data-driven augmented TABS control strategy in a smart building for the south-facing offices, in: ASHRAE and SCANVAC HVAC Cold Climate Conference, 2023.
[44] Union, I.T., *Y.4000 : overview of the internet of things*. 2016.
[45] T. Domínguez-Bolaño, et al., An overview of IoT architectures, technologies, and existing open-source projects, Internet of Things 20 (2022) 100626.
[46] L. Babun, et al., A survey on IoT platforms: communication, security, and privacy perspectives, Comput. Netw. 192 (2021) 108040.
[47] A. Al-Fuqaha, et al., Internet of Things: a survey on enabling technologies, protocols, and applications, IEEE Commun. Surv. Tutorials 17 (4) (2015) 2347–2376.
[48] J. Guth, et al., A detailed analysis of IoT platform architectures: concepts, similarities, and differences, in: B. Di Martino, et al. (Eds.), Internet of Everything: Algorithms, Methodologies, Technologies and Perspectives, Springer Singapore, Singapore, 2018, pp. 81–101.
[49] C.O. Chinedu, A.-H. Amin, M. Emmanuel, A review of energy efficiency strategies in smart buildings: integrating occupant comfort, HVAC optimisation, and building automation, Res. Rev. Sustain. 1 (1) (2025) 48–60.
[50] Y. Himeur, et al., AI-big data analytics for building automation and management systems: a survey, actual challenges and future perspectives, Artif. Intell. Rev. 56 (6) (2023) 4929–5021.
[51] A.J. Khabbazi, et al., Lessons learned from field demonstrations of model predictive control and reinforcement learning for residential and commercial HVAC: a review, Appl. Energy 399 (2025) 126459.
[52] G.P. Henze, K.J. Kircher, J.E. Braun, Why has advanced commercial HVAC control not yet achieved its promise? J. Build. Perform. Simul. 18 (2024) 217–228.
[53] S. Yang, et al., Model predictive control with adaptive machine-learning-based model for building energy efficiency and comfort optimization, Appl. Energy 271 (2020) 115147.
[54] S. Taheri, A.J. Amiri, A. Razban, Real-world implementation of a cloud-based MPC for HVAC control in educational buildings, Energy Convers. Manage. 305 (2024) 118270.
[55] G. Gao, J. Li, Y. Wen, DeepComfort: energy-efficient thermal comfort control in buildings via reinforcement learning, IEEE Internet of Things J. 7 (9) (2020) 8472–8484.
[56] T. Derek, J. Clements-Croome, What do we mean by intelligent buildings? Autom. Constr. 6 (5) (1997) 395–400.
[57] A.H. Buckman, M. Mayfield, S.B.M. Beck, What is a Smart building? Smart Sustainable Built Environ. 3 (2) (2014) 92–109.
[58] J.M. Paul Wellener, H. Ashton, G. James, Smart buildings: Four considerations For Creating People-Centered smart, Digital Workplaces, Deloitte, 2018.
[59] SIEMENS, *smart buildings: striving for the perfect place*. 2019.
[60] Microsoft, *Smart buildings: from design to reality*. 2020.
[61] Bosch. *Smart buildings that benefit people and the environment*.
[62] M. Felderer, et al., Investigating research software engineering: toward RSE research, Commun. ACM 68 (2) (2025) 20–23.
[63] I. Belcic, C.S. *What is AI orchestration?* 2025; Available from: https://www.ibm.com/think/topics/ai-orchestration#:~:text=Greater%20scalability%20One%20of%20the%20primary%20concerns,the%20appropriate%20resources%20in%20the%20right%20places.
[64] Corbusier, L., *Towards a new architecture*. 1923.
[65] M. Iqbal, An Introduction to Solar Radiation, Elsevier Science & Technology, Saint Louis, 1983.
[66] NOAA Global Monitoring Division, *General Solar Position Calculations*. n.d., NOAA Earth System Research Laboratories.
[67] Reno, M., C. Hansen, and J. Stein, *Global horizontal irradiance clear sky models : implementation and analysis*. 2014.
[68] C. Gueymard, Critical analysis and performance assessment of clear sky solar irradiance models using theoretical and measured data, Solar Energy 51 (2) (1993) 121–138.
[69] P. Ineichen, R. Perez, A new airmass independent formulation for the Linke turbidity coefficient, Solar Energy 73 (3) (2002) 151–157.
[70] Remund, J., et al., *Worldwide linke turbidity information*. 2003.
[71] M.A. Atwater, J.T. Ball, Effects of clouds on insolation models, Solar Energy 27 (1) (1981) 37–44.
[72] US Department of Commerce, N. *Forecast terms*. N.d. [cited 2020 December 17, 2020]; Available from: https://www.weather.gov/bgm/forecast_terms.