20% to 80%. Besides ASHRAE 55-2017, ISO 7730 can also be used for de ning ranges [9].

Background Research

ere have been several previous research projects and case studies of the application of BIM for FM including those for BAS (or BMS) and the integration of its real-time data into BIM models. A BIM-FM framework requires the exchange of information between BIM models and dispersed information systems.

Understanding the framework of BIM for FM

e value of BIM application in FM lies in that BIM improves the current manual process of information handover, contributes to more accurate and accessible FM data, and more e cient work order execution Kassem et al. 2015. However, BIM for FM faces challenges ranging from methodologies, shortage of knowledge about implementation requirement and BIM expertise in FM industry [10]. In addition, people do not know enough about the value of BIM for FM

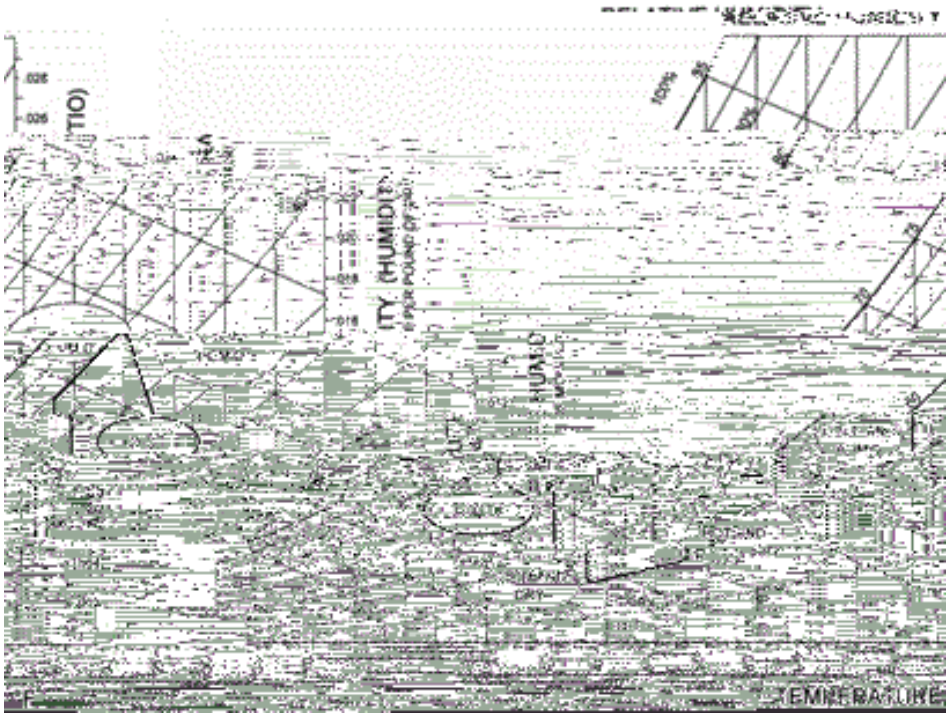

Figure 2: Types of “faults” explained with a psychometric chart.

of schedules, elevations and sections, 3D models and renderings. In this case, Revit worked as a hub of information from other systems, and it also provided reliable source of various presentations of information.

Case study #2: Carnegie Mellon University: Researchers from Carnegie Mellon University developed an integrated multi-view visualizer for interfacing HVAC information to support troubleshooting of HVAC-related problems. In this visualizer developed with Java, users are exposed to a total of eight modules that collects information from a wide range of information systems. e visualizer integrated oor plan view (d), 3D model view (f) and schematic diagram view with text annotations of sensor data from BAS (e). is combination provides clear representation of the location of HVAC components and helps with the real-time monitoring of their working status. By selecting any components from the list (c), information [(d) (e) and (f)] of the corresponding components will be displayed. e visualizer

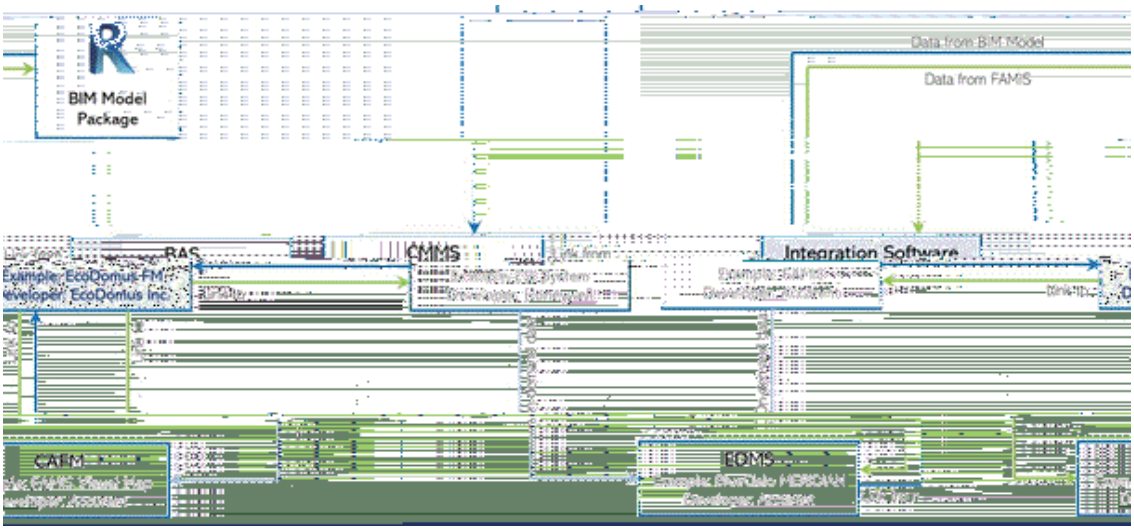

Figure 3: BIM-FM framework adopted by USC FMS.

Figure 4: The location of an inline fan and maintenance manual in EcoDomus.

in a residential building [16]. Data retrieved from re sensors were utilized for establishing a re alarm management system to determine the authenticity of re alarms and to prevent disturbance because of false alarms [17].

e integration of sensor data with BIM is not con ned to indoor environments. An example of such integration is the application of RFID and GPS sensors for building a fusion model that estimates locations of tools, equipment and materials in construction projects [18]. A second example is the use of load sensors in a crane navigation system where the position of li ed objects are displayed in the context of a building and surroundings. Meanwhile, real-time data is collected through sensors and video cameras [19].

Sensor data: Sensor readings are a series of data ordered by a particular rule, mostly by time. e structure of a reading could vary

of the sensors. Another research project focused on using light sensors to control shades and louvers in a building information model through the use of Dynamo rather than the Revit API [27]. BIM+sensor data is an expanding research area; these are just a few examples. None of the examples read showed the link between both comfort and fault visualization in BIM.

Fault detection algorithms

Fault detection has successfully played its role in many engineering domains including automotive and industrial manufacturing for decades, but its application in AEC industry is still under development [28]. Real-time data in buildings need to be monitored to ensure the proper function of facilities and equipment. At the same time, however, detecting anomalies among data sets and problematic behaviors associated with them is also indispensable. Fault detection is achieved by developing algorithms. Some of them are quite simple and straightforward, such as statistical generation model (SGM) based on logarithm likelihood and Adaptive-neuro fuzzy inference system (ANFIS) based on squared prediction error (SPE) index. However, these algorithms sometimes may lead to incorrect results. Advanced algorithms using machine learning or arti cial intelligence can also be used.

Statistical generation model (SGM): Di erent strategies and algorithms for fault detection and alarm generation were proposed by past researchers. A straightforward implementation of fault detection is realized by taking advantage of a "statistical generation model (SGM)" proposed in a research related to BAS. First, a model is constructed based on normal behavior. en, parameters in this normal behavior model is optimized with a maximum likelihood algorithm. With this optimized model, incoming values are examined and compared to this model. "Logarithm likelihood" or "log-likelihood" is the key to the SGM model since it determines whether a value is normal or faulty. Every value in the SGM model has its log-likelihood of appearance. A value will be considered as faulty if this log-likelihood falls below a speci c threshold [29].

Adaptive-neuro fuzzy inference system (ANFIS): Another method of fault detection is called "Adaptive-neuro fuzzy inference system (ANFIS)", in which neural network learning algorithms and fuzzy reasoning are used. In ANFIS method, a squared prediction error (SPE) index is invented to detect sensor faults by calculating the deviances of measured values from predicted values [30]. When ANFIS method is used, SPE index can be calculated with the following formula:

$$\$\%\&'(=* \ _m - \ _p *)$$

where stands for measured values and for predicted values.

To test the fault detecting ability of ANFIS method, researchers introduced a complete sensor fault of 100 μg/m^3 to a PM10 sensor from the 80 th sample. Test results showed that the SPE index increased quickly and exceeded the threshold value marked by the red line. is result indicates that large deviances of measured values from predicted values happened starting the 80 th sample, and the sensor is not working properly. It can also be observed that the SPE index values uctuates intensely as soon as the fault was introduced, with some values fall back to below the threshold value. ese values a er the 80 th sample and below the line are incorrect results, because they indicate that the sensor is working properly, which is contradictory to the fact. erefore, a limitation of this method is the accuracy of results.

Machine learning algorithms: Besides SGM and ANFIS method, there are also a great variety of more advanced fault detection algorithms. ese advanced algorithms are more robust and accurate, and usually involve the application of machine learning and/or arti cial intelligence. ere is an algorithm that features the "prediction-correction" process, during which an updated estimation of system state is provided every time a new measurement is available [28] In addition, machine learning can be combined with a probabilistic model of prediction produced as a result of Gaussian process regression to detect malfunctions in HVAC systems [31].

Simple algorithms: A fault detection algorithm does not always have to be complex to be useful. It just has to be accurate within the bounds of its context. For example, a null reading from a temperature sensor in a room is also a fault as a value is expected. Determining the reason for the null reading might be complex, but the algorithm (check for null readings) is not. A second example is checking for a value within a speci ed range. If the value is not in the range, it is a fault. For example, a temperature set point might be 72°F (22.2°C) and the oat 3°F (20.6°C to 23.9°C) to achieve comfortable working conditions. A temperature reading of 80°F (26.7°C) would be considered a fault in this case, but perhaps not for another set of conditions/ranges.

Methodology- Adafruit io Reader Tool

Revit allows users to add new features to its existing functions. ese

Citation: Kensek K (2020) A BIM-based Visualization Tool for Facilities Management: Fault Detection Through Integrating Real-Time Sensor Data into BIM. J Archit Eng Tech Res 9: 228.

On this feed information page, all feed IDs and feed keys for current user are available. Every feed in Adafruit IO is assigned a unique feed key that can be used to check real-time data in this feed.

https://io.adafruit.com/api/v2/{username}/feeds/{feed key}/

Citation: Kensek K (2020) A BIM-based Visualization Tool for Facilities Management: Fault Detection Through Integrating Real-Time Sensor Data into BIM. J Archit Eng Tech Res 9: 228.

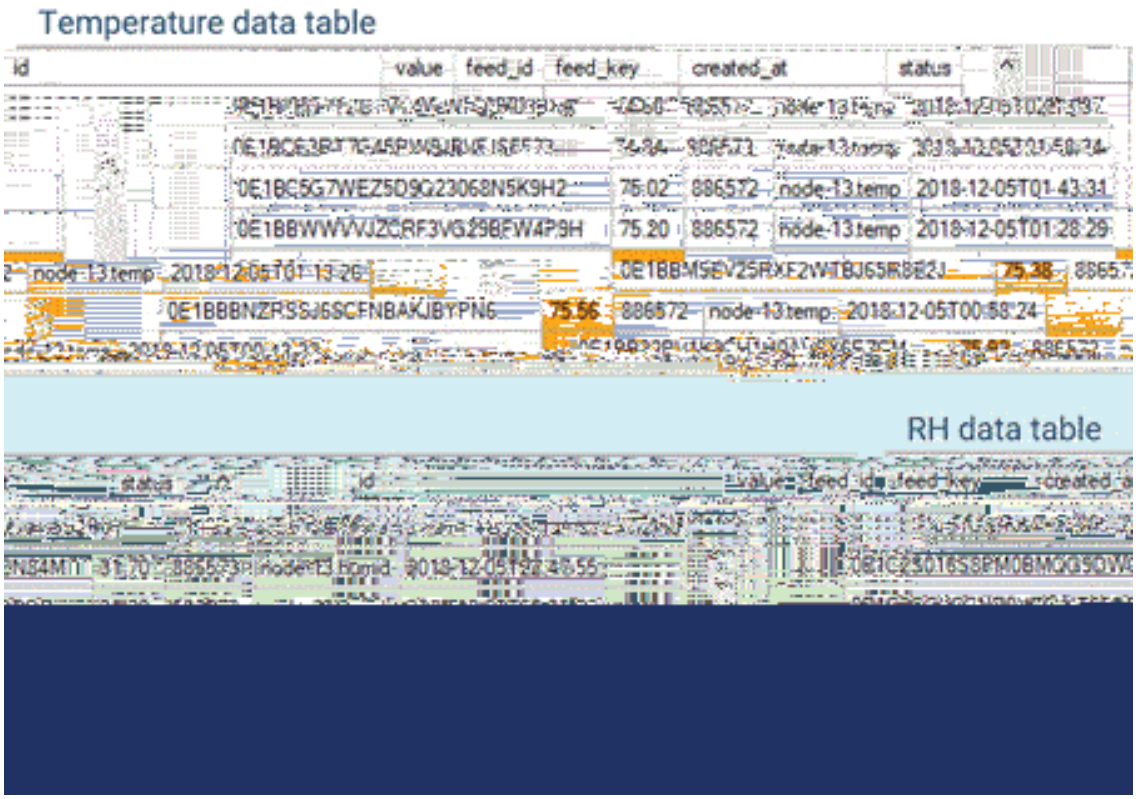

Figure 11: Data values out of range with color schemes Orange-higher than the maximum; light blue- lower than the minimum.

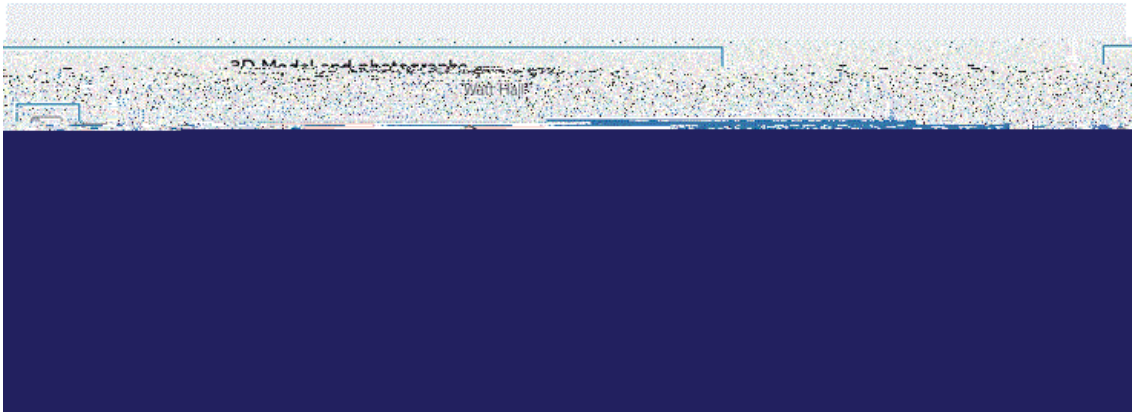

Figure 12: Sensor model visualization.

Running the macro brings up the highlighted sensor model in a 3D view and the main interface of Adafruit IO Reader with real-time sensor data retrieved from the server (Figure 17).

In the main interface of the Adafruit IO Reader, most of the space on the main interface window is used for data table view panels and line plot view panels visualizing sensor data. e interface is currently comprised of 12 functional modules (Figure 18).

- **Menu bar items:** Menu commands related to synchronizing sensor data with Adafruit IO server, clearing data display on the interface, editing thermal comfort zone and le path for storing sensor data les.
- **Sensor list:** A dropdown list containing ve IoT sensors installed in Watt Hall. By selecting any sensor from the list, the program automatically load temperature and RH data les and display them in view panels. At the same time, the program takes users to the selected sensor model and zoom in to the view of that sensor.
- **Revit element ID:** Every sensor model in Revit is assigned a

Figure 13: Locations of sensor installation in Revit and their photographs.

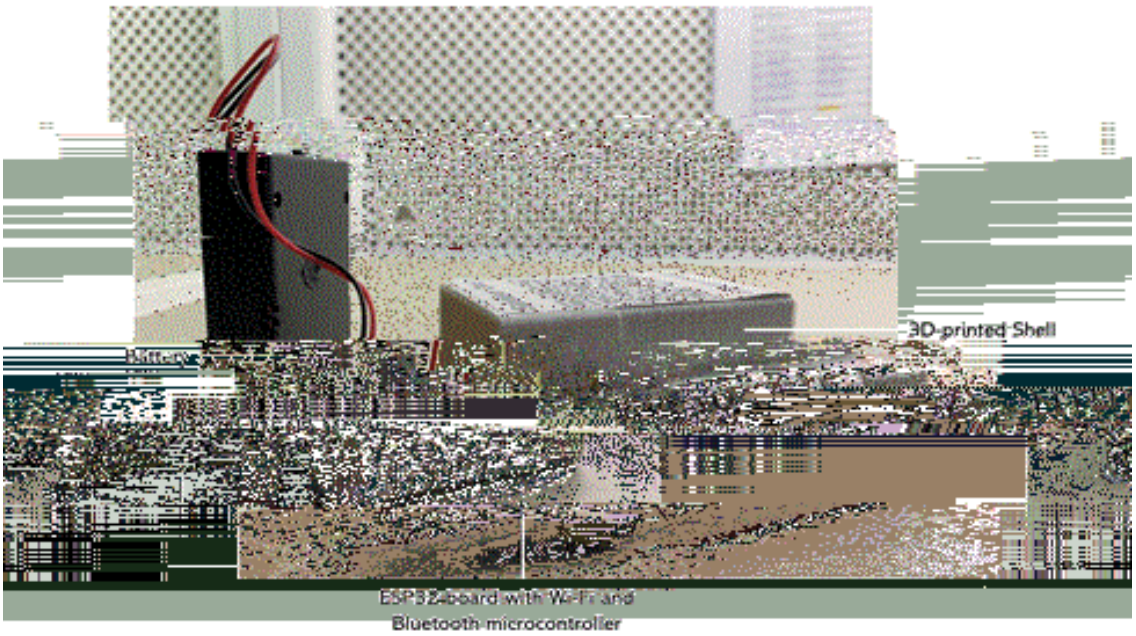

Figure 14: An IoT sensor.

six-digit Revit element ID. is is used in source code to create the many-to-one relationship between data feeds and a sensor model.

- **Update data:** A shortcut button for synchronizing sensor data with data feed stored on Adafruit IO server.
- **Reset lter:** A temporary button for developer's testing and debugging only. is will be removed in future versions.
- **Temperature tab control:** Switch between temperature data table and line plot.
- **Temperature view panel:** Display temperature data table (by default) and line plot.
- **RH tab control:** Switch between RH data table and line plot.
- **RH view panel:** Display RH data table (by default) and line plot.
- **Room photo:** Display the photo of the room where the sensor is installed.
- **Sensor photo:** Display the close-up view of the sensor.
- **Display settings:** e settings related to manipulating data

Figure 15: Sensor locations in Watt Hall.

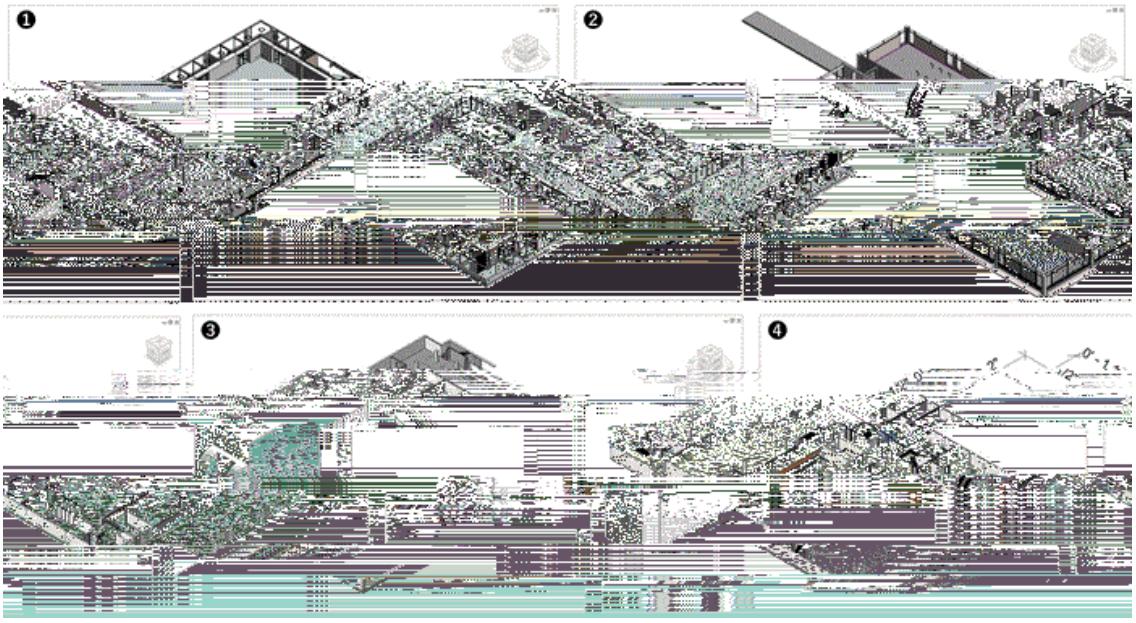

Figure 16: Watt Hall 3D models (1-3) and IoT sensor model (4).

tables such as lter values according to start and/or date, or the status of values (faulty or normal).

Data retrieval, pruning, and synchronization: Sensor data is not automatically synchronized with the latest version on the server. It must be done manually either from menu bar or the shortcut button on the interface. e average time spent on data syncing in the tool is approximately 8 seconds per time, so it is not ideal to make tool users wait by synchronizing data every time the program is launched.

Save customized settings: e bidirectional interaction between the con guration le and Adafruit IO Reader can be used for customized comfort zones. e program reads the customized setting from the le and displays the setting in the Comfort Zone window, where users are free to adjust the threshold values for new settings.

Data tables and plots: Data tables and line plots take most of the space on the interface, and they serve as the basis of other program features. Data tables in Adafruit IO Reader share the same layout as those CSV les that can be opened with Excel upper le (Figure 19). e most recent 1,000 data values are available in both data tables. is

Figure 17: The result of running macro “A_Watt_Hall_Building_Science_Corner”.

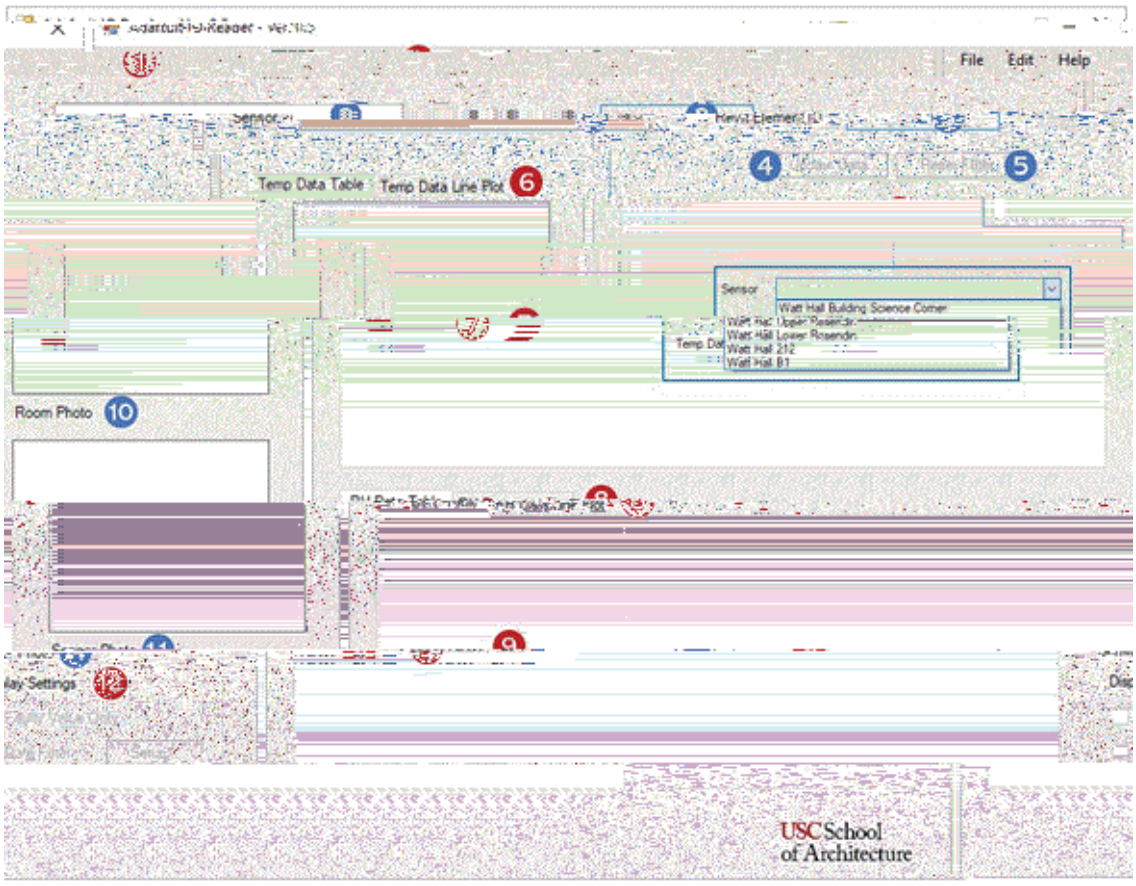

Figure 18: The main interface.

is the amount of data for approximately 15 days. Past values older than 15 days will be erased and replaced by new values to dispose space for storage. Line plots, as a more visualized presentation, are generated based on data values in the second column upper right (Figure 19). Users can switch between tables and graphs with tab controls at the upper le corner of view panels. Both data tables and line plots can be manipulated by di erent display settings to allow users to focus on a speci ed portion of data.

Full and partial tables/plots: Manipulation of data tables is completed by "Display Settings" on lower right corner of the main interface. One of the settings is named "Date Filter." Sometimes, there may be a time range of interest where data values need special attention. Adafruit IO Reader provides a date lter that allows users to focus on only an excerpt from the full table. Either start date or end date, or both dates can be used for setting a range. e lter works on not only to data tables, but also line plots. With such a lter, the line plot view looks like "zoomed in", enabling the observation of data uctuation in a more legible manner with a smaller scale.

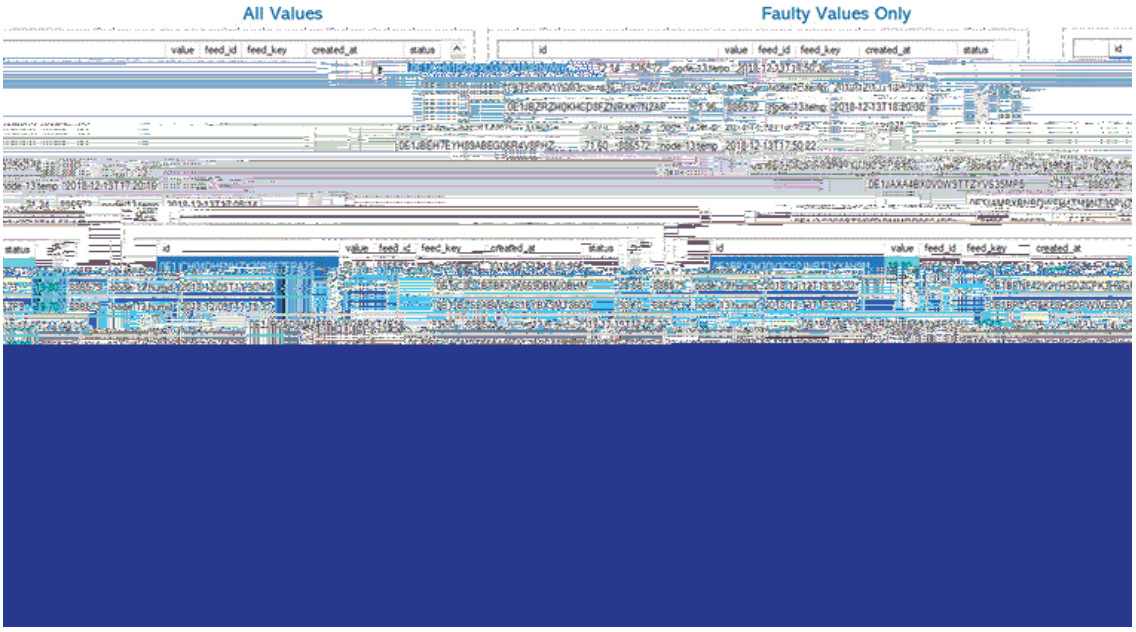

Figure 20: All values and faulty values (Upper left -All values; Upper right-faulty values only; Lower-Steps and clicks).

Figure 21: 3D model and information for sensor in Watt Hall Lower Rosendin.

Adafruit IO Reader is composed of nine main features separated into three categories as previously mentioned. ese features are examined with the case study of Watt Hall, and most of the features performed satisfactorily by always producing desired outputs, except for some minor issues with the navigation to sensor models. Of particular note is the integration of the data within a building information model. FM personnel can use Revit to check the sensor list, discover the location of the sensors in the digital model and by the room photographs linked to each sensor, and show the data as tables or line plots. Settings can be created to de ne a "fault." In the case study, ASHRAE 55 and ISO 7730 were used to set high and low values for temperature and relative humidity. If the room values were outside of these settings, the sensors would be highlighted allowing facilities managers immediate feedback to uncomfortable conditions. A fault is also detected if there is not a value ("nan") being sent by the sensor.

Limitations

Despite that the nine main features worked properly, there is still

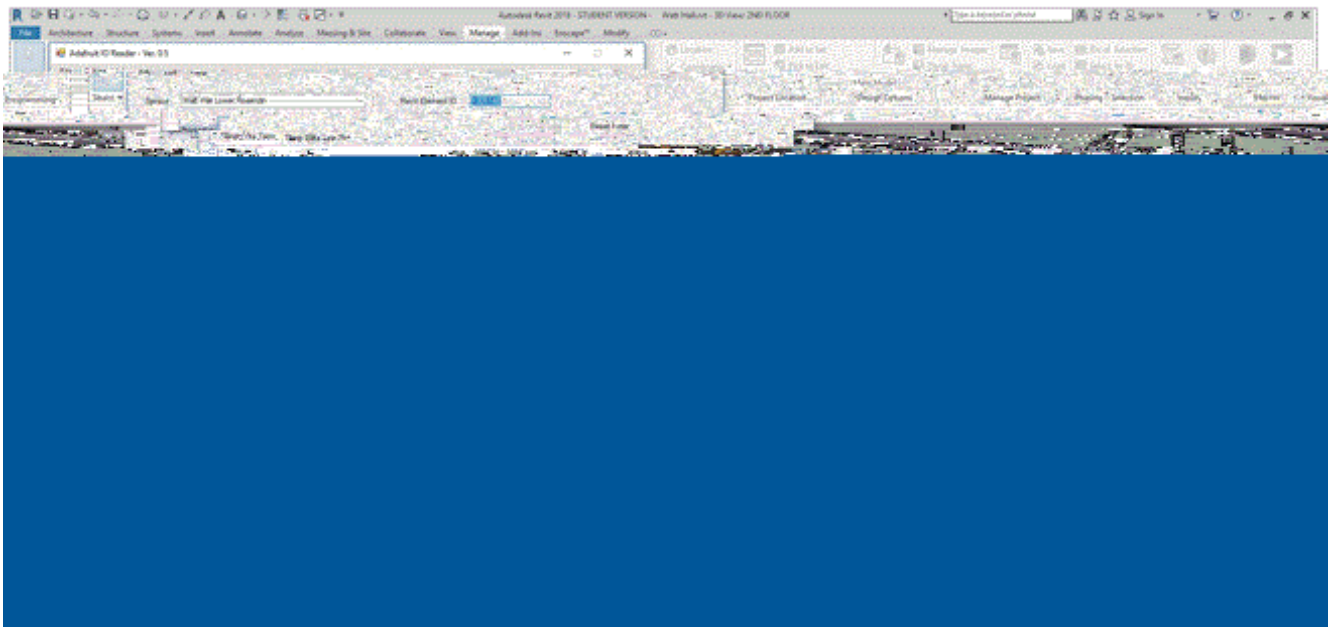

Figure 24: The integration of sensor data and BIM through Adafruit IO Reader.

plenty of room for improvement for the Adafruit IO Reader. e biggest limitation lies in the installation of IoT sensors and the in uence on availability of data feeds. Amongst the ve sensors in Watt Hall, only one at Building Science Corner was plugged in, while another four are powered by AA batteries. A possible explanation for doing so is that the wiring between sensors and sockets will reveal the locations of sensors and expose them to the risk of being taken away. Taking advantage of solar energy is ideal in terms of sustainability, but it is impractical in indoor settings without daylighting [33]. Given that AA batteries were used for powering most of the sensors in Watt Hall, the replacement of batteries becomes a serious issue. If the replacement stops, the sensors will eventually out of power and stop collecting data. Another problem derived from the previous one is the limitation of the data feed pruning algorithm. e current pruning algorithm is solely designed for sensor data les generated by ASAIR IoT sensors and retrieved from Adafruit IO servers. In case of processing data les generated by other types of sensor, it is likely that some necessary information may be erroneously pruned, which will lead to data les with disordered formatting. A third limitation is that the sensor management is accessible only through adding Revit Family (RFA) les of sensor model to Watt Hall model and through editing source code. ese approaches are far too complicated for users new to Revit and C#, and lacks e ciency when a large amount of sensor models and corresponding information need to be added to or deleted from the model. Adafruit IO Reader currently collects a total of 10 data feeds (both temperature and RH feeds) for ve sensors.

Managing sensors and data feeds of this amount is still possible by manual editing of codes, but a brand-new and automated approach for sensor management should be taken into account before Adafruit IO Reader can deal with more sensors.

Future work

Future work includes addressing existing limitations and expanding the functionalities of Adafruit IO Reader. e main areas of work include improving the speed of the work ow, updating the code, allowing the use of data feeds from other sources, and imbedding the tool in existing facility management platforms. e synergy of features from these platforms and Adafruit IO Reader will provide FM managers with more comprehensive information about facilities and equipment in a building, supporting both quotidian monitoring work and, when necessary, corrective maintenance work (troubleshooting). Another alternative is to instead create the ability to make custom fault ranges in existing so ware like Dasher 360 to provide the same functionality.

Conclusion

e combination of visualized sensor data and sensor models were evaluated by a case study of Watt Hall, and the result showed that Adafruit IO Reader is able to achieve the integration by navigating users to sensor models, while providing corresponding sensor data information that is open to customized manipulation through various display settings. e ndings of the research is of value for FM managers in terms of supporting with visualized user interfaces decision-making processes especially related to HVAC systems and thermal comfort. Adafruit IO Reader could be a prototype for developing more sophisticated products that help the data management in FM, or it could be integrated into currently used FM platforms to better streamline work ows with more exhaustive information during regular management work and troubleshooting work. e speci c contributions of this research is that the display of sensor data, de nition of “fault” (for comfort-de ned by ASHRAE 55 and ISO 7730), and links to the sensors’ locations occur directly within the BIM so ware.

Acknowledgement

This research did not receive any specifc grant from funding agencies in the public, commercial, or not-for-profit sectors.

References

1. McGraw-Hill (2019) National BIM Standard-United States. APEC Meeting Document Database. Accessed.
2. Hosseini MR, Roelvink R, Papadonikolaki E, Edwards DJ, Pärn, E (2018)

Integrating BIM into facility management: Typology matrix of information handover requirements. International Journal of Building Pathology and Adaptation 36: 2-14.

3. Matarneh ST, Danso-Amoako M, Al-Bizri S, Gaterell M, Matarneh R (2018) Developing an interoperability framework for building information models and facilities management systems. In Creative Construction Conference pp: 1018-1027.
4. Sensor analytics (2019) IoT Agenda. Accessed.
5. Liu X, Akinci B (2009) Requirements and evaluation of standards for integration of sensor data with building information models. J Comput Civil Eng pp: 95-104.