

Article

Integration with 3D Visualization and IoT-Based Sensors for Real-Time Structural Health Monitoring

Hung-Fu Chang *  and Mohammad Shokrolah Shirazi

R. B. Annis School of Engineering, University of Indianapolis, Indianapolis, IN 46227, USA; shirazim@uindy.edu
* Correspondence: hchang@uindy.edu; Tel.: +1-317-788-5018

Abstract: Real-time monitoring on displacement and acceleration of a structure provides vital information for people in different applications such as active control and damage warning systems. Recent developments of the Internet of Things (IoT) and client-side web technologies enable a wireless microcontroller board with sensors to process structural-related data in real-time and to interact with servers so that end-users can view the final processed results of the servers through a browser in a computer or a mobile phone. Unlike traditional structural health monitoring (SHM) systems that deliver warnings based on peak acceleration of earthquake, we built a real-time SHM system that converts raw sensor results into movements and rotations on the monitored structure's three-dimensional (3D) model. This unique approach displays the overall structural dynamic movements directly from measured displacement data, rather than using force analysis, such as finite element analysis, to predict the displacement statically. As an application to our research outcomes, patterns of movements related to its structure type can be collected for further cross-validating the results derived from the traditional stress-strain analysis. In this work, we overcome several challenges that exist in displaying the 3D effects in real-time. From our proposed algorithm that converts the global displacements into element's local movements, our system can calculate each element's (e.g., column's, beam's, and floor's) rotation and displacement at its local coordinate while the sensor's monitoring result only provides displacements at the global coordinate. While we consider minimizing the overall sensor usage costs and displaying the essential 3D movements at the same time, a sensor deployment method is suggested. To achieve the need of processing the enormous amount of sensor data in real-time, we designed a novel structure for saving sensor data, where relationships among multiple sensor devices and sensor's spatial and unique identifier can be presented. Moreover, we built a sensor device that can send the monitoring data via wireless network to the local server or cloud so that the SHM web can integrate what we develop altogether to show the real-time 3D movements. In this paper, a 3D model is created according to a two-story structure to demonstrate the SHM system functionality and validate our proposed algorithm.

Keywords: structural health monitoring; Internet of Things; 3D modeling; real-time system



Citation: Chang, H.-F.; Shokrolah Shirazi, M. Integration with 3D Visualization and IoT-Based Sensors for Real-Time Structural Health Monitoring. *Sensors* **2021**, *21*, 6988. <https://doi.org/10.3390/s21216988>

Academic Editor: Dimitrios Chronopoulos

Received: 29 August 2021

Accepted: 15 October 2021

Published: 21 October 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Important civil structures, such as bridges, energy utilities, nuclear power plants, and dams, require regular monitoring of their behaviors to support management decisions, reduce the loss of lives under natural disasters. Moreover, as a result of monitoring behaviors, necessary maintenance can be handled in time to minimize hazardous impacts on a large number of resources that are commonly employed within the constructions [1]. Structural health monitoring (SHM) aims to develop a system that can continuously check a structure, deliver current structural responses, and even alert when the structure exceeds the design limit. A structure's behaviors often vary according to its age, usage, or environmental factors. SHM systems in the past research are designed with pre-assigned or specific types of sensor deployments due to the pre-defined calculation or in-device computations. Some SHM studies examined the monitoring approaches by evaluating

the result afterward without investigating the possible application in real-time. However, when a natural disaster happens (e.g., earthquakes or gusty wind), an immediate reaction according to the observation on structural behaviors is critical. These rapid responses rely on real-time structural monitoring. Hence, a more effective SHM should intuitively visualize the structural real-time behaviors from the monitoring results across platforms and equip the flexibilities to adjust the computation with responding to the change in the sensor deployment, such as adjusting the number of devices or adapting the sensor location or sensor replacement. With the advancement of the micro-electro mechanical system (MEMS), wireless and computing technologies, combining sensors, micro-controller boards, client-side scripts, and internet servers to build a real-time cross-platform access-anywhere SHM system becomes more feasible.

A typical SHM system includes sensors, data processing components, and health evaluation units [2]. Recent technology development in these three areas has been shifted to utilize the latest standards and software libraries. For example, In the area of sensors or sensor networks, the traditional wireless sensor network within a local and close network (i.e., intranet) has been moved to the internet [3]. Various sensors attached to the micro-controller board, called the Internet of Things (IoT), can be easily programmed to interact with the servers on the internet. This new wireless sensor network based on the Internet of Things often involves web and database servers or even more complex computing infrastructure, such as the edge or cloud. The communication between IoT devices and servers often uses standard communication protocols, such as Hypertext Transfer Protocol (HTTP) or Message Queue Telemetry Transport (MQTT), to interact with each other and to communicate server systems across platforms over the internet [4,5]. IoT devices are referred to smart objects or things since they commonly offer a webpage-like user interface, which is implemented during the development phase, for their users to interact with them via mobile applications or internet browsers. These new wired and wireless sensor devices can also cooperate with multiple devices and different servers, unlike those in the past SHM research, which only focused on the computation and warning mechanism of a single device [6–9]. Instead of studying the monitoring and warning mechanism on a single IoT device, we would like to further include the consideration of the complex interactions among sensor devices, servers on the internet, and the user's clients in the SHM system. Then, the SHM system migrates from a single server or personal computer within a local network to one that can integrate local computing servers, edge computing nodes, and various types of servers distributed across multiple regions on the cloud. At the same time, the design and development of this new SHM system also bring many challenges to node-to-node communication, data processing, algorithm, sensor deployment, and system architecture.

To show visualized structural behaviors and movements that can be used in design or study in the future, we built up a sensor device similar to other IoT smart things to collect monitoring data in our SHM system development. Our sensor device embeds a website-like user interface that can allow users to directly access the device and configure its device-server connection, deployment location at global coordinate, and sampling rate. Then, every sensor delivers real-time three-dimensional linear acceleration which will be translated to the displacement at global coordinate by the server in real-time. To display the actual movement of each part of the structure in a 3D model, the global displacement data at each sensor location need to be converted to each structural element's displacement and rotation at its local coordinate. To incorporate a plug-and-play feature for sensor devices in our SHM, we create a key-value data format for keeping a generic element-to-element relationship. An algorithm is invented for handling this conversion. To process the enormous amount of data generated by the devices, we design a system architecture that can deal with this in real-time. This paper is an initial study, which continues our previous work related to IoT sensor research, toward our ultimate goal: a real-time SHM system that handles big data and heterogeneous sensors in an edge-cloud computing environment.

2. Related Works

Earlier SHM research studied a method or a sensor device to issue an early warning when the external force exceeded the design capacity of the structure or the implementation of the closed sensor network for monitoring [2,3,10–15]. Recent works focused on structural damage detections. Aloisio et al. [16] conducted a variance-based sensitivity analysis of various damage indicators to understand their possible usage limits. Zhoi and Wahab [17] incorporated the cosine measure and modal assurance criterion (MAC) indicators in the application of transmissibility-based structural damage detection. Sony and Sadhu [18] used various numerical simulations to validate the synchrosqueezing transform (SST) for progressive damage assessment. In applying IoT in the SHM, Muttillio et al. [19] built an IoT sensor system that integrated ADXL355 sensors, a Same3X8E ARM Cortex-M3 microcontroller, and a personal computer via a RS485 communication bus. Their suggested system evaluated the damage indicator but does not send monitoring data wirelessly to a server on the internet. Abdelgawad and Yelamarthi [20] used a USB Wi-Fi module, a Raspberry Pi 2 board, buffers, ADCs, DACs, and PZTs to build an IoT platform. The results will be sent to the server hosted in ThingWorx. Martinez et al. [21] created an IoT-based prototype that uses a Raspberry Pi 3 board, a USB wireless card, and accelerometer sensors to show the acceleration values in a x-y chart on a server. Aba et al. [22] developed an IoT-based sensor device that could monitor petroleum pipelines and showed the results in real-time on the ThingSpeak website.

3. Materials and Methods

3.1. Sensor Device

Our design choice emphasizes a device that can easily be installed in the existing structure and built with inexpensive costs. Hence, we selected the NodeMCU (a microcontroller board with a 32-bit ESP8266 chip), the ADXL345 accelerometer sensor [23], and a MicroSD board (see Figure 1). This integration almost occupied all the general purpose input/output (GPIOs) that NodeMCU provides (i.e., ten 3.3 voltage digital GPIOs and one 1.8 voltage analog input GPIO). In addition to the inexpensive cost for accelerometer sensors, the ADXL345 provides a better detection range and preciseness among MMA7455, MMA8451, MPU6060, MMA7660, and ADXL335 chips. The micro-SD board is not essential but it saves the monitoring results locally as backup, and the backup data can be analyzed offline if needed.

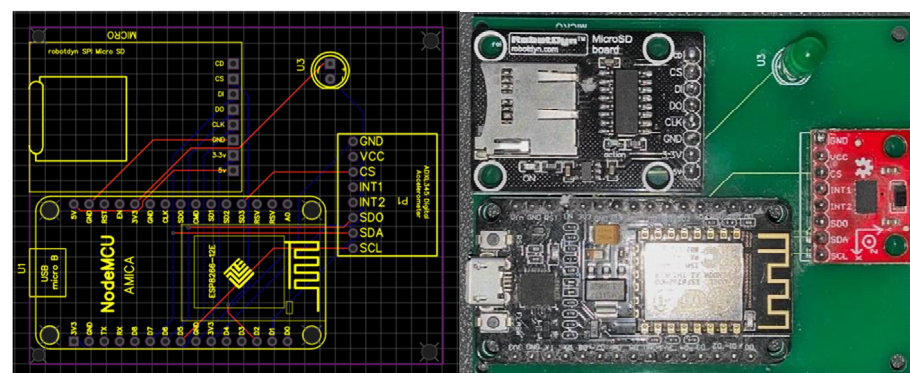


Figure 1. Schematic design and circuit board of the sensor device.

The wireless capability is provided by the ESP8266 chip. A small amount of data can be saved in its flash memory. The ESP8266 has a station infrastructure (STA) mode that sets a service set identifier (SSID) to allow other computers to connect to its wireless network; this means that the device links to the wireless network and obtains a local IP address during STA mode. Inside the device, we stored several web pages in the ESP8266 chip's built-in flash memory, letting users save configuration variables inside the flash memory. Because the user interface is a web page, the users can use a browser to set

the key variables (e.g., the wireless network's SSID, account and password, server host, etc.). During the initialization stage of the device, a universal unique identifier (UUID) is generated and saved in the flash memory. The server uses this UUID to associate with its deployed location, the relation with other devices, and the moving point in the 3D model. When real-time monitoring records associated with the device's identification number are received by the server, the overall movement of the 3D structural model is calculated according to our algorithm and shown to the client.

It is critical to ensure sufficient power to a sensor device to monitor the structure in real-time. The micro USB port in the NodeMCU provides the entire device power. Using the micro USB port extends the sensor device's power source options (e.g., using a replaceable high-capacity battery) and increases the device's portability. On the other hand, the device buffers the monitoring data in the built-in flash memory and then pushes the data via the network to an internet server. All the computations are performed in the server. Decreasing the number of data transmission times to the server and shifting the computation from the sensor device to the server should reduce the power consumption.

3.2. System Architecture

Modern software widely applies various web technologies to facilitate inter-operability between programs or to exchange data among servers. Due to the recent advance of Javascript technology, particularly in the server-side run-time environment (i.e., Node.js) and client-side framework and libraries, software can enjoy the benefits from the Javascript; that is, its scalability, high-performance, and asynchronous non-blocking executions. Hence, our server software integrates various Javascript server-side and user interface libraries—Node.js, SocketIO, Three.js, and C3.js (see Figure 2). All the server-side Javascript components run on the Node.js environment. The SocketIO is for real-time communication, which sends bi-directional messages according to the pre-defined events, between the client side (i.e., user's browser) and the server side. So, each axis' displacement values as well as the transformation and movement data of each 3D element are embedded in the server-to-client message sent via the protocol created by the SocketIO on both sides. The Three.js on the client-side page uses WebGL to provide application programming interfaces (APIs) for users to create and render the 3D model. We create an online 3D model editor and a real-time display based on the Three.js. Since Three.js can run on any browser for online 3D model rendering and editing, the 3D model can be displayed in any client [24]. The C3.js is the chart library to show real-time acceleration values on the X-Y line chart.

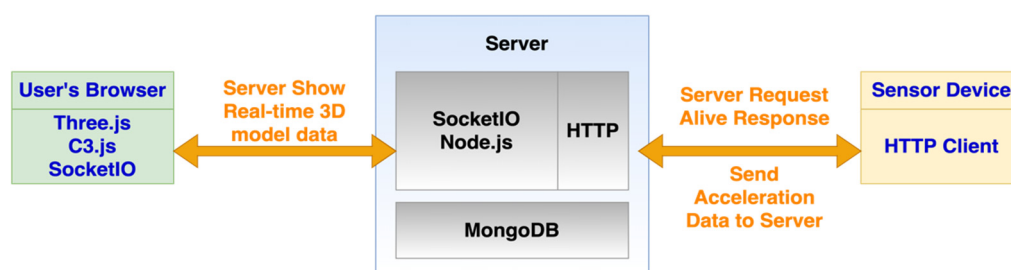


Figure 2. Architecture of the SHM system.

The software architecture of our server is layer style. The upmost layer is HTTP, responsible for the request and response between sensor device and server. All the IoT sensor device monitoring results are pushed to this server according to its sampling frequency via HTTP protocol. All the required computations are in between the HTTP and data storage layers. Inside the computation part, each monitoring point's global displacements are calculated from monitored acceleration values and then the global displacements can be converted into each 3D structural element's rotation and shift. Regarding the data storage, the system uses MongoDB, which is classified as a document-oriented (NoSQL) database.

There will be two different kinds of usages of the SHM system (see Figure 3). Firstly, the 3D model is formed according to the real-world civil structure and then all sensor device details (e.g., locations and identification number) are scaled down to the model. All the model and sensor details are saved in the database. When the monitoring starts, all real-time monitoring results collected from IoT sensor devices are sent to the server via the wireless network. To reduce the transmission data, the monitoring results are firstly sent to the local server. The local server then sends replica and computation results to the cloud server. During this monitoring phase, once the user's client accesses any server, the user can see the 3D model moving in real-time.

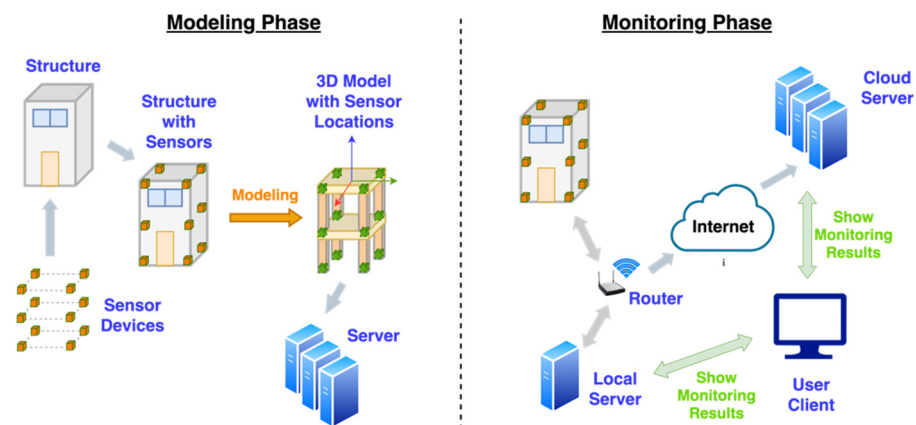


Figure 3. Two different types of the SHM usages.

3.3. Three-Dimensional Structural Model Visualization

3.3.1. Challenges and Assumptions

The real-world structural displacement combines the deformation of each element (e.g., bending or shrinking) and the ground movement. The actual deformation of the structure depends on its material, shape, and force. After the structure is built, the composition of the material is often unknown. That is why most SHM sensor devices only monitor acceleration and issues warnings when the monitoring acceleration is close to the designed value. The nature of having many unknown factors of the to-be-monitor after-built structure particularly brings tough challenges to our system for displaying each structural element's actual deformation and displacement on its 3D model according to three-dimensional monitored displacements.

The overall displacements derived by the sensor's monitoring results are the combinational effects of horizontal and vertical forces. Traditional systems use forces and material stress and strain model to derive the displacements of the structural element. One assumption has to be made to treat the monitored displacements as an aftereffect of complex combinational forces and only use displacements to illustrate the structure's 3D model movement. We assume that every column section between two monitoring points is a rigid body—no deformation. Therefore, the assumption overlooks the initial linear and later non-linear behaviors of most construction materials. We attempt to overcome this weakness by putting more sensors on one column. If four corners of a floor are monitoring points that are supported by columns, the deformation of the floor can be decided by the displacements of the column at each corner.

Our primary focus is on horizontal displacements. We limit our problem to horizontal measurements, which implies that the vertical (i.e., z-axis) measurement is neglected and is regarded as a shift to the entire structure. Because horizontal forces mainly cause a structure's horizontal displacement, the contribution of the vertical force on the horizontal displacement can be neglected.

3.3.2. Structural Global Displacement

The displacement can be derived from the numerical integration of the acceleration. In other words, the area under the graph of the velocity curve over time is the displacement. Previous studies [25–28] have suggested methods to calculate it from the accelerometer so we applied those integration equations to get the global displacement at each sensor location.

Getting global displacement at each monitoring point is the first step to visualize the 3D model movements. However, the global displacement is insufficient to understand how each connected structural element moves (e.g., how floor and column moves) because elements in the 3D model rotate and move at their local coordinate. As a result, we must figure out a way to translate the global displacements into each element's rotation and displacement at its local coordination system.

3.3.3. Conversion from Global Displacement to Element's Local Rotation and Displacement

Earthquakes produce three-dimensional ground forces. The displacements and rotations of the structure can be viewed as the result of these three ground forces. Figure 4 shows that the movements of the two connected columns (i.e., Column01 and Column12) are caused by the horizontal forces from the position at time t_0 to another one at t_1 . Every P_{ij} point is a sensor monitoring location, and signals the point at the i -th position at the j -th step. To illustrate how to derive global-to-local conversion equations under this situation, we divide the movement between these two timestamps into six steps. The first three steps are displacements along with the Y_G , X_G , and Z_G axes, and the latter three steps are rotations at X_L , Y_L , and Z_L axis. Since all six steps are caused by forces along with X_G and Y_G , there is not any rotation at Z_L axis.

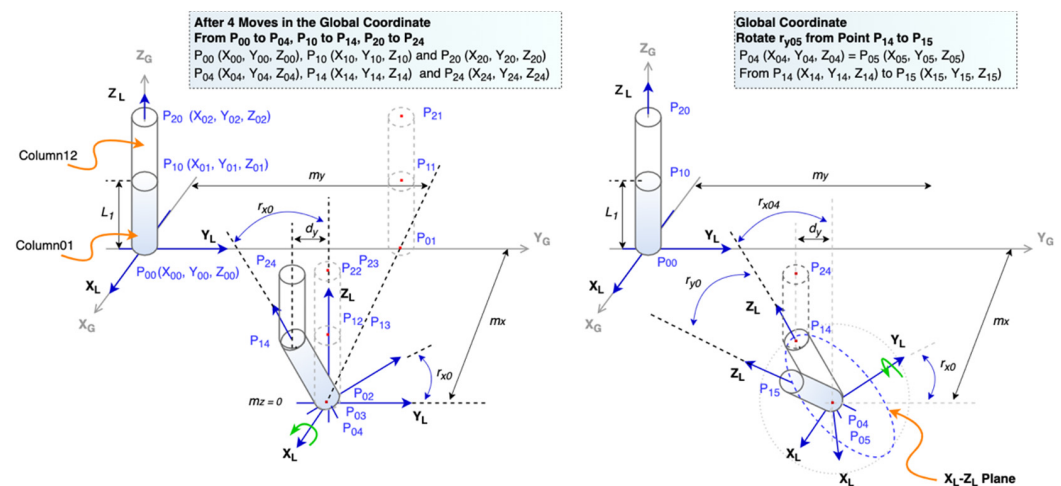


Figure 4. Movements caused by X-Y forces.

From Figures 4 and 5, we can derive Equations (1) and (2) where L_1 is the length of the Column01, $X_1(t_1)$ and $Y_1(t_1)$ are coordinates of the top point of the Column01 at time t_1 , $X_0(t_1)$ and $Y_0(t_1)$ are coordinates of the bottom point at Column01 at time t_1 , r_{y0} is the angle at Y_L axis, and r_{x0} is the angle at X_L axis.

$$r_{y0} = ((X_1(t_1) - X_0(t_1)) / L_1) \quad (1)$$

$$r_{x0} = ((Y_1(t_1) - Y_0(t_1)) / (L_1 \cos r_{y0})) \quad (2)$$

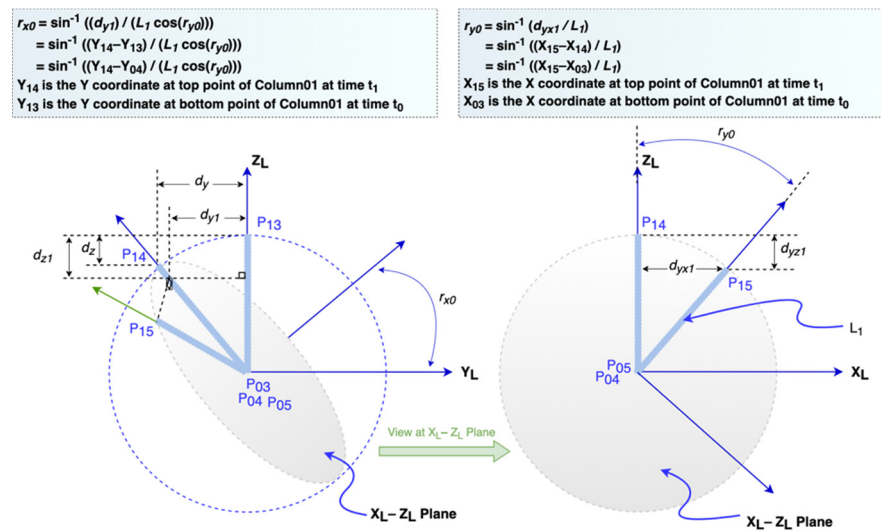


Figure 5. The rotation r_{x0} and r_{y0} at the Column01.

Both rotations will also contribute z-axis displacement d_{z1} . The value at time t_1 can be established after r_{y0} and r_{x0} are computed.

$$d_{z1} = L_1(1 - \cos \cos r_{y0} \cos \cos r_{x0}) \quad (3)$$

$$Z_1(t_1) = L_1 \cos \cos r_{y0} \cos \cos r_{x0} \quad (4)$$

When we have multiple connected columns and two rotations are applied at a particular column n , its two rotation angles and z-axis displacement can be calculated from the generalized equations as follows:

$$r_{yn-1} = ((X_n(t_i) - X_{n-1}(t_i)) / L_n) \quad (5)$$

$$r_{xn-1} = ((Y_n(t_i) - Y_{n-1}(t_i)) / (L_n \cos \cos r_{yn-1})) \quad (6)$$

$$d_{zn} = L_n(1 - \cos \cos r_{yn-1} \cos \cos r_{xn-1}) \quad (7)$$

$$Z_n(t_i) = L_n \cos \cos r_{yn-1} \cos \cos r_{xn-1} \quad (8)$$

where L_n is the length of the n -th column, $X_n(t_i)$ and $Y_n(t_i)$ are coordinates of the top point of the n -th column at time t_1 , $X_{n-1}(t_i)$ and $Y_{n-1}(t_i)$ are coordinates of the bottom point of the n -th column at time t_i , r_{yn-1} is the angle at Y_L axis, and r_{xn-1} is the angle at X_L axis.

However, 3D models are designed to rotate at the center of the object, unlike the above scenario—rotating at the bottom of each column. To translate the rotation in Figure 5 to the rotation at the bottom, the 3D model element must include a shift after it rotates at the same required angle. Given that there are multiple columns connected, at the n -th column, we have to consider two types of shifts. The first one is from the displacement at its bottom point, which is caused by the $(n-1)$ -th element's shift. The bottom point's shift is measured at time t_i to t_{i-1} and its movement is in three directions so each direction's displacement can be calculated by the difference of coordinate value at each axis. Therefore, the bottom point displacement at the x -axis is $X_{n-1}(t_i) - X_{n-1}(t_{i-1})$ and the same principle can be applied to the y and z -axis. The second type of movement caused by the rotation is illustrated in Figure 6. For the n -th column rotating around the x and y -axis, the total displacements are shown as follows:

$$d_{cyn-total} = (L_n/2)(1 - \sin \sin r_{xn-1}) + (Y_{n-1}(t_i) - Y_{n-1}(t_{i-1})) \quad (9)$$

$$d_{cxn-total} = (L_n/2)(1 - \sin \sin r_{yn-1}) + (X_{n-1}(t_i) - X_{n-1}(t_{i-1})) \quad (10)$$

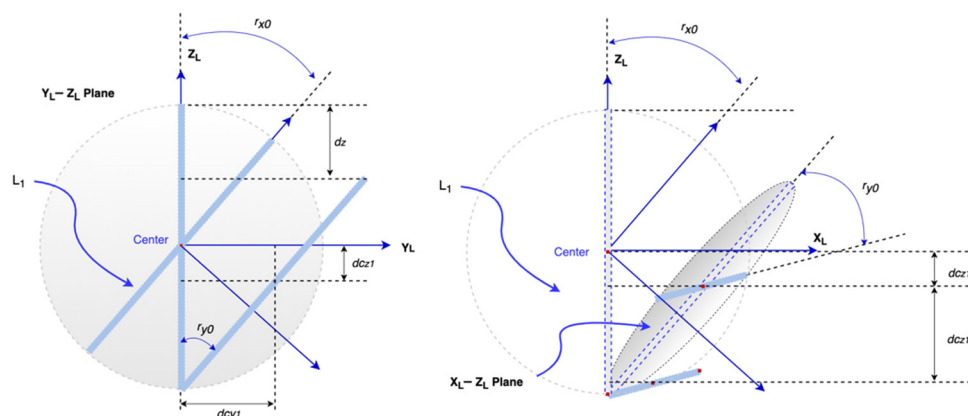


Figure 6. Shifts after two rotations.

The displacement at the z-axis is the sum of the center point declining introduced by both rotations (i.e., $d_{czn} + d'_{czn}$) and also the bottom point's movement.

$$d_{cyn-total} = d_{czn} + d'_{czn} = (L_n/2)((1 - \cos \cos r_{xn-1}) + (1 - \cos \cos r_{yn-1})) + (Z_{n-1}(t_i) - Z_{n-1}(t_{i-1})) \tag{11}$$

The global coordinate of the center point measured at the time t_i is in the following equation.

$$(X, Y, Z) = \left(\frac{X_n(t_i) + X_{n-1}(t_i)}{2}, \frac{Y_n(t_i) + Y_{n-1}(t_i)}{2}, \frac{Z_n(t_i) + Z_{n-1}(t_i)}{2} \right) \tag{12}$$

We can apply the same idea to derivat the conversion from global displacement to the local beam element's rotation and displacement. For a beam parallel to the y -axis, its rotations can be at the z - or x -axis (see Figure 7).

$$r_{xn-1} = ((X_n(t_i) - X_{n-1}(t_i)) / L_n) \tag{13}$$

$$r_{zn-1} = ((Z_n(t_i) - Z_{n-1}(t_i)) / (L_n \cos \cos r_{xn-1})) \tag{14}$$

$$d_{yn} = L_n(1 - \cos \cos r_{xn-1} \cos \cos r_{zn-1}) \tag{15}$$

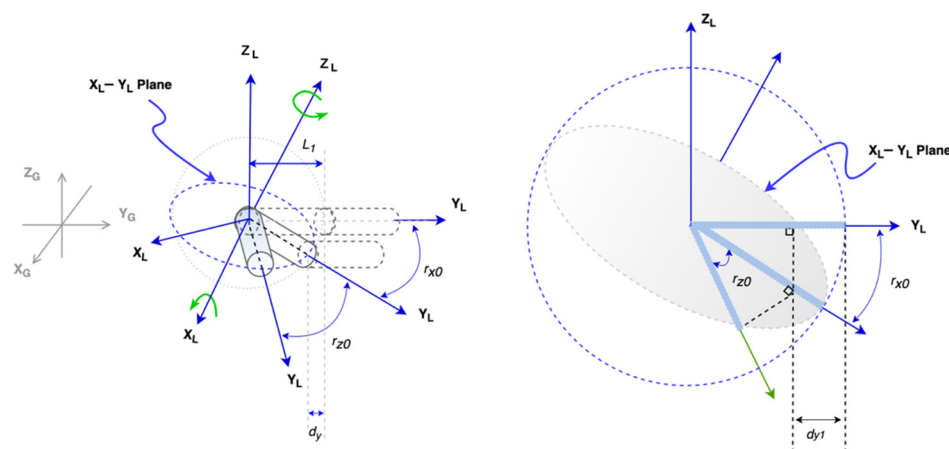


Figure 7. Two rotations of the beam.

Similarly, for a beam parallel to the x -axis, when its rotations are in the z - or y -axis, the equations for getting the angles are Equation (16) and Equation (14), respectively.

$$r_{yn-1} = ((Y_n(t_i) - Y_{n-1}(t_i)) / L_n) \tag{16}$$

$$d_{xn} = L_n (1 - \cos \cos r_{yn-1} \cos \cos r_{zn-1}) \tag{17}$$

3.3.4. Sensor Deployment

Past research [29–31] already discussed the importance of IoT device deployment. In our system, the sensor location is particularly critical because it determines the details of the overall monitored structure’s transformation and movement. Sensors are installed at the monitoring points. The sections between two sensor devices are treated as connected columns or beams. As a result, when a structure is modeled, its model must compose of separated elements, where both ends of the element are the monitoring points. Adding sensors to columns and joints must happen before attaching them to beams. By adding more sensor devices to each structural element (i.e., beam or column), the non-linear deformation can be shown. Figure 8 shows that the more detailed deformation can be displayed while the number of sensors increases. We must know that the minimum sensor installations are at the joints on the columns. Otherwise, our above equations cannot be adequately applied for displaying movements in our SHM system.

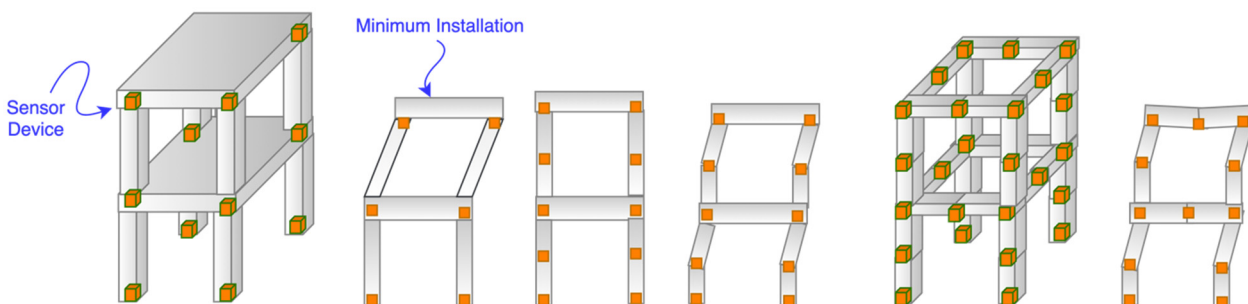


Figure 8. Sensor deployment.

3.3.5. Timestamp

Timestamp or time is critical to the real-time system. The sensor device only contains a timer that measures time in milliseconds starting from zero when it boots. The timestamp keeps increasing until it shuts down. Our way to get time is to map the timestamp from its timer to the server’s clock time by sending a request and receiving the server’s response for inquiry and recording the time. Figure 9 shows that the timestamp x_1 should be mapped to the server’s clock time t_1 . Due to latency, when the server records the clock time tr_1 associated with x_1 , the time already passed dtr_1 . Hence, the mapped server time cannot be accurate.

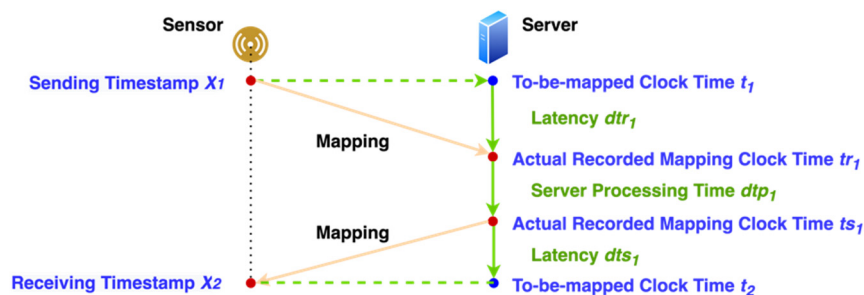


Figure 9. Mapping between sensor device local timestamp and server clock time.

We try to calculate the average latency and use it to adjust the clock time so we can get the recorded mapping time closer to the one that should be mapped. The following equations show how we map timer x_1 and x_2 to clock time t_1 and t_2 , respectively.

$$\text{Average Latency } dt_1 = ((x_2 - x_1) - (ts_1 - tr_1))/2 \quad (18)$$

$$x_1 \rightarrow t_1 \cong tr_1 - dt_1 \quad (19)$$

$$x_2 \rightarrow t_2 \cong ts_1 + dt_1 \quad (20)$$

According to the equations above, mapped clock time in each sensor device is an approximate value. Therefore, it is not easy for all devices to get the acceleration at the same clock time. What is worse is that multiple devices receive different network latencies. So, the best strategy is to reduce the latency. As a result, having a local server to communicate with all the servers becomes a better solution.

3.3.6. Data Format

To use our suggested method to display the overall movement of the structure in its 3D model, the SHM system needs to save more than measurement data. Sensor deployment information and relation between different types of structural elements, such as floors, columns, and beams, are also required. For example, to calculate the displacement of a column, all the monitoring points at both ends of the connected parts are needed.

All the relations are organized in key-value pairs and arrays in the JavaScript Object Notation (JSON) format. Figure 10 shows an example of how the data are saved.

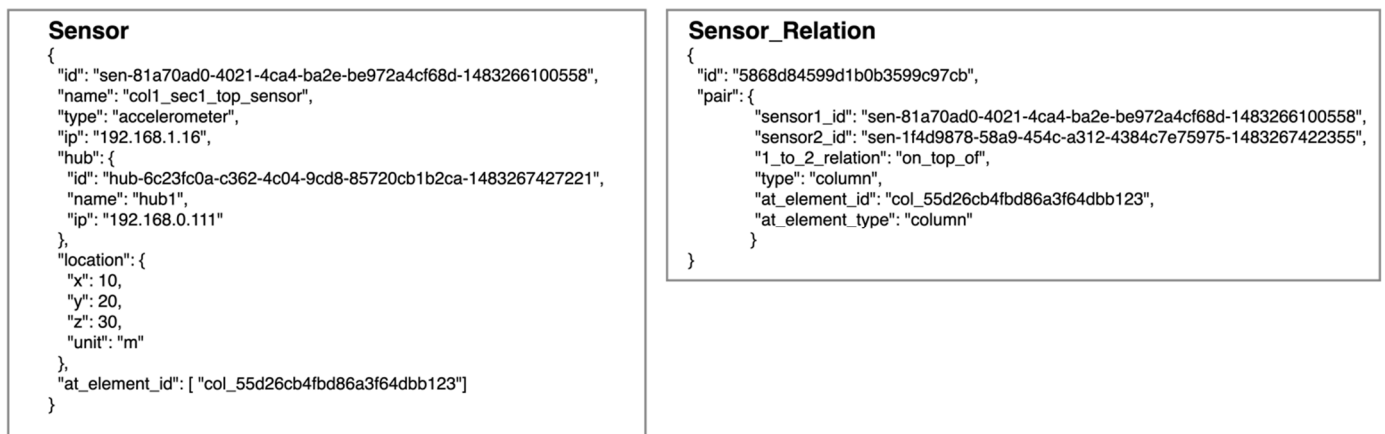


Figure 10. Example of sensor and sensor relation data.

The structural 3D model is created after the user edits it on the website. The model data saved in the JSON format contain camera information, geometries, materials, and objects. After the monitoring data are converted into each object's rotation, displacement, or even transformation, the dynamic behaviors under external forces can be shown in its 3D model.

4. Implementation and Validation

4.1. Implementation

4.1.1. Sensor Device Software

The sensor device's user interface is a tab-like web page (see Figure 11). The "Setup" tab allows users to connect to a wireless network so that the device can acquire an IP address from the router (see Figure 12). Users configure the server IP and hostname in the "Cloud" Table After all the required information is set, the "Home" tab shows its IP and identification number. The "Home" tab provides the real-time acceleration measurements at x, y, and z directions after the enable button is clicked. Users can also select a different

sample rate. The buffer size input is used to specify the internal memory buffer for caching the monitoring data. As for the “Setup” tab, user account and password can be set in order to protect illegal access to this user interface of this sensor box. In this way, saved configurations will not be altered.

Figure 11. Home tab of sensor device user interface.

Figure 12. Cloud and setup tabs of sensor device user interface.

4.1.2. Server Software

The implementation contains three major parts: (1) project, structural model, and sensor information; (2) 3D model building tool; and (3) 3D model real-time movement display. Users can create a project that includes various structures. Sensor device locations and identifications are saved in the server. After the user creates a 3D model in our application, the model data will automatically be saved in the server for later display. The last part is the real-time view for 3D model movement. The user can move the camera position or zoom in/out the view to see the movement from different angles.

Figure 13 shows the 3D model editor in our server application inside our SHM system. All the structural elements are built by using the “box” object. The Three.js client-side component can allow users to use any browser to edit their 3D models according to the to-be monitored structure. Because of using a browser, the feature of editing anywhere is realized. We also want to emphasize why global to local conversion is necessary in real-time display. In Figure 14, the local coordinate on each structural element is at its center. It is important to note that the displacement, rotation, and transformation are all at their local coordination system instead of the global coordinate system. Each element’s local coordination system will change after its local rotation or shift. It introduces the challenges of global to local conversion.

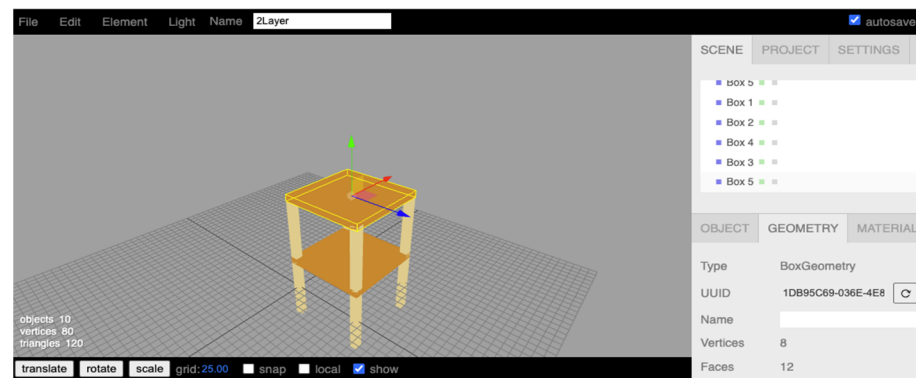


Figure 13. Two-story structure on 3D model editor of the SHM system.

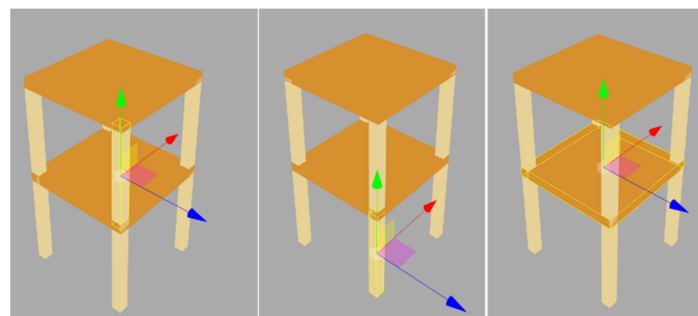


Figure 14. Local coordination at each element.

4.2. Validation

Our validation targets correctly display real-time 3D movements according to the structural monitoring data sent by multiple sensor devices. Unsynchronized monitoring timestamps across different sensor devices or inaccurate global and local conversion equations will cause the detached structural model parts (e.g., a beam disconnects with a column) or unreasonable movement of elements (e.g., a clockwise rotation becomes an incorrect counterclockwise rotation).

There are two parts in the validation. First, we test the time adjustment in the sensor device. Four devices are put on the small-scale shaking table and shook with a maximum acceleration of 0.2 g. The monitored acceleration values will be examined at each timestamp. The second part is a simulation for investigating our conversion approach, which is a step before we later conduct a scaled structure experiment on a shaking table. We build a two-story 3D model for displaying the movement. On the model, the locations of the sensors are at each joint so we use eight sensor devices to send the simulation data to the server on the local network. The movements of the 3D model will be checked via a browser that accesses the server. Figure 15 shows the procedure of how we deal with the conversion equation validation.

4.2.1. Time and Latency

Since four devices are moved at the same acceleration, after our time adjustment for each device, all of them are expected to get the same acceleration value at the same timestamp. This implies that their wave pattern should be very close to each other. According to Figure 16, the time adjustment is accurate because it shows four monitoring curves are very close. Table 1 shows the standard deviation of the acceleration values of the four devices at each timestamp. A very low maximum standard deviation also validates our time adjustment method.

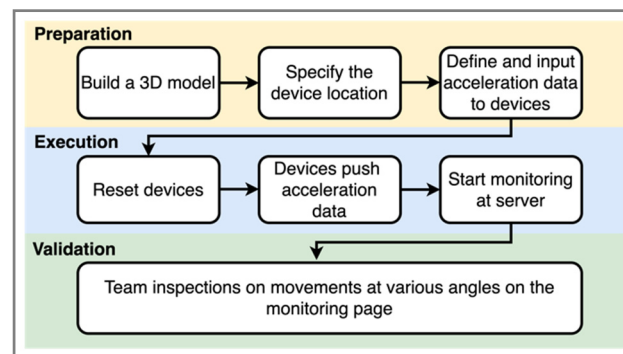


Figure 15. Global to local conversion validation procedure.

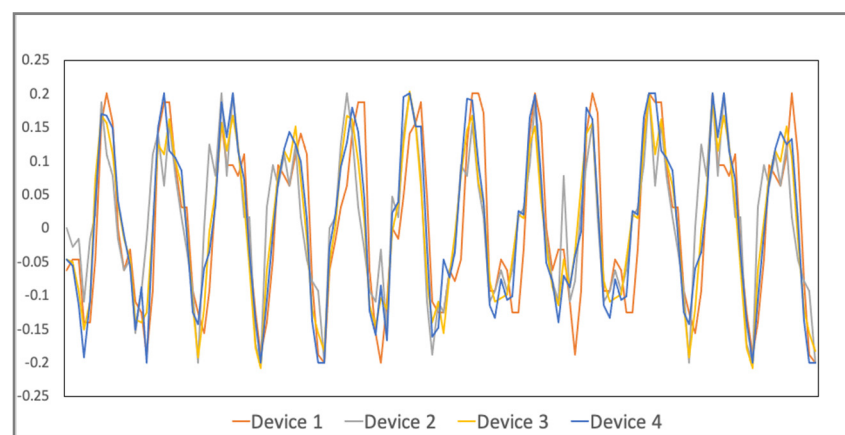


Figure 16. Acceleration monitoring of four sensor devices.

Table 1. Standard deviation of accelerations at each timestamp.

Maximum Standard Deviation	Minimum Standard Deviation
0.0950045	0

4.2.2. 3D Movement Calculation

Our server for hosting the real-time SHM system is implemented in the one core CPU, 512MB ram, and Ubuntu OS machine on the local network. The two-story structural 3D model (see Figure 13) is built to validate our proposed global to local conversion. Every sensor is assigned at each joint so eight devices are used in the simulation. The acceleration data pushed by the sensors continuously contribute the displacements at the sampling rate of 20 Hz to the server. It is noteworthy that the monitoring sample time interval is different from the display time interval. In our validation, we use one second. This can avoid a display lag on the client's browser, which is caused by too frequent updates and heavy client-server data traffic.

Two criteria were used to validate the global to local conversation. At each joint, connected elements cannot detach during every movement because our conversion equations are under the assumption that there are no broken joints and rigid body rotations and displacements. The other criterion examines rotations or displacements in the correct direction. Three people inspected the movements by using their browser at the same time, checked the model movements from various angles by moving the camera, and then made observation notes for each movement. Figure 17 shows that eight movements repeat the same sequence continuously because the sensor devices keep delivering a series of pre-defined acceleration values. Throughout the whole simulation, no disjoints between elements or wrong rotations were seen by the inspectors. This indicated that the system

displayed movements correctly and validated our conversion equations. Because a successful 3D model movement also relies on synchronous monitoring time and accurate movement interpolation, correct 3D model movements also indirectly validate our time synchronization and interpolation methods. These validations are performed through the simulation, which prepares our SHM system to reproduce the actual movements of the to-be-monitored structure.

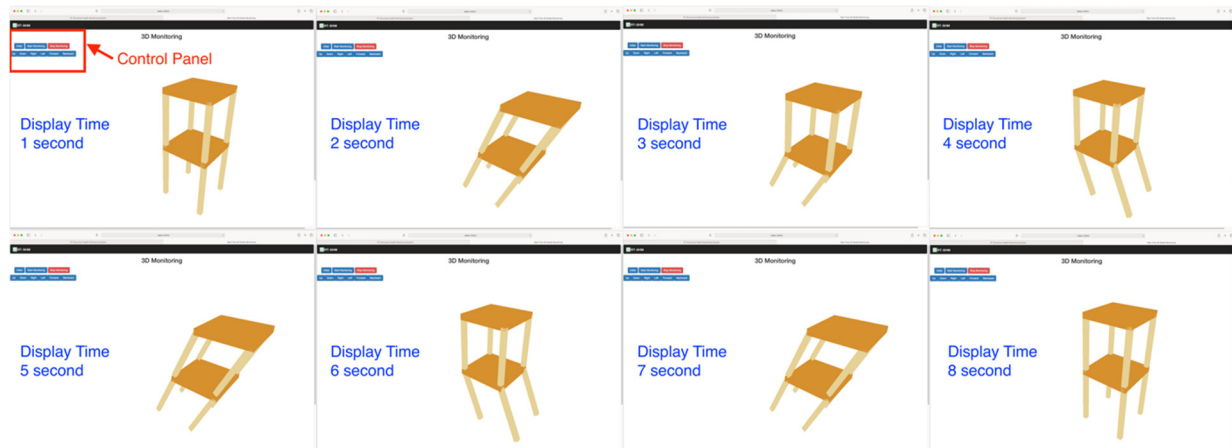


Figure 17. Two-story model movements.

5. Discussion and Conclusions

In this paper, we proposed a SHM system that can demonstrate real-time structural movements in a 3D model on the website so that users can view monitoring results by using any client device that has browsers. In our SHM system, the users can build up their structural 3D model according to the real-world structure and save all the sensor information. With the model and sensor data, once the real-world structure moves under the external force, the user can see the real-time movements on the display webpage.

Our validation results showed that our conversion method, monitoring time synchronization approach, and displacement adjustment equation could display the overall behaviors correctly. Our current result is the initial step for accurately displaying real-world structural monitoring results in a 3D model. That means, we have to consider elastic and non-linear behaviors of the structural element. We understand that our conversion equations have limitations due to our rigid body assumptions. Real-world structural behaviors under external forces do not merely act as a rigid body. Although we can use more sensors to display bent structural elements approximately, we will need to study a 3D deformation equation to show the non-linear behaviors of every structural element in the future. We also know that using acceleration values to get displacements might not be ideal. However, we argue that this deficiency does not impact our proposed approach. In other words, once we can apply a more accurate displacement measurement technique, our SHM can display more realistic 3D movements of the to-be monitored structure.

Author Contributions: Conceptualization, H.-F.C.; methodology, H.-F.C.; software, H.-F.C.; validation and analysis, H.-F.C.; writing—original draft preparation, H.-F.C.; writing—revise, review and editing, M.S.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Chong, K.P. Health Monitoring of Civil Structures. *J. Intell. Mater. Syst. Struct.* **1998**, *9*, 892–898. [[CrossRef](#)]
2. Sun, M.; Staszewski, W.J.; Swamy, R.N. Smart sensing technologies for structural health monitoring of civil engineering structures. *Adv. Civ. Eng.* **2010**, *2010*, 724962. [[CrossRef](#)]
3. Mainetti, L.; Patrono, L.; Vilei, A. Evolution of Wireless Sensor Networks towards the Internet of Things: A Survey. *J. Comput. Des. Eng.* **2015**, *2*, 148–156.
4. Atzori, L.; Iera, A.; Morabito, G. The Internet of Things: A Survey. *Comput. Netw.* **2010**, *54*, 2787–2805. [[CrossRef](#)]
5. Sundmaeker, H.; Guillemin, P.; Friess, P.; Woelfflé, S. (Eds.) *Vision and Challenges for Realising the Internet of Things*; European Commission-Information Society and Media DG: Brussels, Belgium, 2010.
6. Krishnamurthi, R.; Kumar, A.; Gopinathan, D.; Nayyar, A.; Qureshi, B. An Overview of IoT Sensor Data Processing, Fusion, and Analysis Techniques. *Sensors* **2020**, *20*, 6076. [[CrossRef](#)] [[PubMed](#)]
7. Buratti, C.; Conti, A.; Dardari, D.; Verdone, R. An overview on wireless sensor networks technology and evolution. *Sensors* **2009**, *9*, 6869–6896. [[CrossRef](#)]
8. Mattern, F.; Floerkemeier, C. From the Internet of Computers to the Internet of Things. *J. Univers. Something* **2005**, *11*, 11–111.
9. Gubbi, J.; Buyya, R.; Marusic, S.; Palaniswami, M. Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions. *J. Comput. Des. Eng.* **2013**, *29*, 1645–1660. [[CrossRef](#)]
10. Masri, S.; Sheng, L.; Caffrey, J.; Nigbor, R.; Wahbeh, M.; Abdel-Ghaffar, A. Application of a web-enabled realtime structural health monitoring system for civil infrastructure systems. *Smart Mater. Struct.* **2004**, *13*, 1269–1283. [[CrossRef](#)]
11. Xu, N.; Rangwala, S.; Chintalapudi, K.; Ganesan, D.; Broad, A.; Govindan, R.; Estrin, D. A wireless sensor network for structural monitoring. In Proceedings of the ACM Conference on Embedded Networked Sensor Systems, Baltimore, MD, USA, 3–5 November 2004.
12. Kim, S.; Pakzad, S.; Culler, D.; Demmel, J.; Fenves, G.; Glaser, S.; Turon, M. Health monitoring of civil infrastructures using wireless sensor networks. In Proceedings of the 6th International Conference on Information Processing in Sensor Networks (IPSN), Cambridge, MA, USA, 25–27 April 2007.
13. Ceriotti, M.; Mottola, L.; Picco, G.P.; Murphy, A.L.; Guna, S.; Corra, M.; Pozzi, M.; Zonta, D.; Zanon, P. Monitoring Heritage Buildings with Wireless Sensor Networks: The Torre Aquila Deployment. In Proceedings of the 2009 International Conference on Information Processing in Sensor Networks, San Francisco, CA, USA, 13–16 April 2009.
14. Rice, J.; Mechtov, K.; Sim, S.; Spencer, B.; Agha, G. Enabling Framework for Structural Health Monitoring using Smart Sensors. *Struct. Control Health Monit.* **2011**, *18*, 574–587. [[CrossRef](#)]
15. Paek, J.; Chintalapudi, K.; Govindan, R.; Caffrey, J.; Masri, S. A Wireless Sensor Network for Structural Health Monitoring: Performance and Evaluation. In Proceedings of the Second IEEE Workshop on Embedded Networked Sensors, Sydney, NSW, Australia, 30 April–1 May 2005; Institute of Electrical and Electronics Engineers: Piscataway, NJ, USA, 2005; pp. 1–10.
16. Aloisio, A.; Di Battista, L.; Alaggio, R.; Fragiaco, M. Sensitivity analysis of subspace-based damage indicators under changes in ambient excitation covariance, severity and location of damage. *Eng. Struct.* **2020**, *208*, 110235. [[CrossRef](#)]
17. Zhou, Y.L.; Wahab, M.A. Cosine based and extended transmissibility damage indicators for structural damage detection. *Eng. Struct.* **2017**, *141*, 175–183. [[CrossRef](#)]
18. Sony, S.; Sadhu, A. Identification of progressive damage in structures using time-frequency analysis. In Proceedings of the CSCE Annual Conference, Montreal, BC, Canada, 12–15 June 2019.
19. Muttillio, M.; Stornelli, V.; Alaggio, R.; Paolucci, R.; Di Battista, L.; Rubeis, T.; Ferri, G. Structural Health Monitoring: An IoT Sensor System for Structural Damage Indicator Evaluation. *Sensors* **2020**, *20*, 4908. [[CrossRef](#)] [[PubMed](#)]
20. Abdelgawad, A.; Yelamarthi, K. Internet of Things (IoT) Platform for Structure Health Monitoring. *Wirel. Commun. Mob. Comput.* **2017**, *2017*, 6560797. [[CrossRef](#)]
21. Martinez, S.; Jiménez, J.; Baculima, R.; Serrano, I. IoT-based Microseismic Monitoring System for the Evaluation of Structural Health in Smart Cities. In Proceedings of the First Ibero-American Congress, ICSC—CITIES 2018, Soria, Spain, 26–27 September 2018.
22. Aba, E.N.; Olugboji, O.A.; Nasir, A.; Olutoye, M.A.; Adedipe, O. Petroleum pipeline monitoring using an internet of things (IoT) platform. *SN Appl. Sci.* **2021**, *3*, 180. [[CrossRef](#)] [[PubMed](#)]
23. Analog Devices; ADXL 345 Datasheet Rev E. 2013. Available online: <https://www.analog.com/media/en/technical-documentation/data-sheets/ADXL345.pdf> (accessed on 1 July 2021).
24. Nyamsuren, P.; Lee, S.; Hwang, H.; Kim, T. A web-based collaborative framework for facilitating decision making on a 3D design developing process. *J. Comput. Des. Eng.* **2015**, *2*, 148–156. [[CrossRef](#)]
25. Seifert, K.; Camacho, O. *Implementing Positioning Algorithms Using Accelerometers*; Application Note; Freescale Semiconductor: Tempe, AZ, USA, 2007.
26. Martin, A.; Manfred, P. Calculation of displacements of measured accelerations, analysis of two accelerometers and application in road engineering. In Proceedings of the 6th Swiss Transport Research Conference, Ascona, Switzerland, 15–17 March 2006.
27. Wei-Meng, N.; Fang, L.; Zi-Yuan, Q.; De-Qing, G. Small Displacement Measuring System Based on MEMS Accelerometer. *Math. Probl. Eng.* **2019**, *2019*, 3470604. [[CrossRef](#)]
28. Andrew, B.; Graham, W.; William, W. Deriving Displacement from 3-Axis Accelerometers. In Proceedings of the Computer Games, Multimedia & Allied Technology, Singapore, 11–12 May 2009.

-
29. Alablani, I.; Alenazi, M. EDTD-SC: An IoT Sensor Deployment Strategy for Smart Cities. *Sensors* **2020**, *20*, 7191. [[CrossRef](#)]
 30. Burhanuddin, M.A.; Mohammed, A.A.-J.; Ismail, R.; Hameed, M.E.; Kareem, A.N.; Basiron, H. A Review on Security Challenges and Features in Wireless Sensor Networks: IoT Perspective. *J. Telecommun. Electron. Comput. Eng.* **2018**, *10*, 17–21.
 31. Garrido-Hidalgo, C.; Hortelano, D.; Roda-Sanchez, L.; Olivares, T.; Ruiz, M.C.; Lopez, V. IoT Heterogeneous Mesh Network Deployment for Human-in-the-Loop Challenges Towards a Social and Sustainable Industry 4.0. *IEEE Access* **2018**, *6*, 28417–28437. [[CrossRef](#)]