Informes de la Construcción
Vol. 66, 536, e044
octubre-diciembre 2014
ISSN-L: 0020-0883
doi: http://dx.doi.org/10.3989/ic.13.151

Integration of Environmental Sensors with BIM: case studies using Arduino, Dynamo, and the Revit API

Integración de sensores medioambientales con BIM: casos de estudio usando Arduino, Dynamo, y Revit API

K. M. Kensek (*)

ABSTRACT

This paper investigates the feasibility of connecting environmental sensors such as light, humidity, or CO_2 receptors to a building information model (BIM). A base case was created in Rhino; using Grasshopper and Firefly, a simple digital model responded to lighting-levels detected by a photoresistor on an Arduino board. The case study was duplicated using Revit Architecture, a popular BIM software, and Dynamo, a visual programming environment, in an innovative application. Another case study followed a similar procedure by implementing the Revit API directly instead of using Dynamo. Then the process was reversed to demonstrate that not only could data could be sent from sensors to change the 3D model, but changes to parameters of a 3D model could effect a physical model through the use of actuators. It is intended that these virtual/physical prototypes could be used as the basis for testing intelligent façade systems before constructing full size mock-ups.

Keywords: Environmental sensors; BIM; building information model; visual scripting; Dynamo; intelligent facades.

RESUMEN

Este estudio investiga la posibilidad de conectar sensores ambientales como de luz, humedad, o dióxido de carbono con un modelo de información de un edificio (siglas BIM en inglés). Un caso base fue creado en Rhino; usando Grasshopper and Firefly, donde un simple modelo digital respondió a niveles de luz detectados por un foto resistor en una tarjeta Arduino. El caso de estudio fue duplicado usando Revit Architecture, una herramienta popular en BIM, y Dynamo, un ambiente de programación gráfica, en una creativa aplicación. Un segundo caso de estudio siguió un procedimiento similar implementando Revit API directamente en vez de usar Dynamo. Entonces el proceso fue revertido para demostrar que no solamente la información podría ser enviada desde sensores para cambiar el modelo tridimensional, pero cambios en los parámetros de un modelo tridimensional podrían afectar un modelo físico mediante el uso de actuadores. Se espera que esos modelos virtuales puedan ser usados como base para probar sistemas de fachadas inteligentes antes de la construcción de modelos físicos de tamaño real.

Palabras clave: *Sensores ambientales; BIM; modelos de información de edificios; codificación virtual; Dynamo; fachadas inteligentes.*

(*) University of Southern California - School of Architecture. Los Angeles, CA (USA)
Persona de contacto/Corresponding author: kensek@usc.edu (K. M. Kensek)

Cómo citar este artículo/*Citation:* Kensek, K. M. (2014). Integration of Environmental Sensors with BIM: case studies using Arduino, Dynamo, and the Revit API. *Informes de la Construcción,* 66(536): e044, doi: http://dx.doi.org/10.3989/ic.13.151.

Recibido/*Received*: 24/10/2013
Aceptado/*Accepted*: 18/02/2014
Publicado on-line/*Published on-line*: 19/11/2014

1. INTRODUCTION

The concept of embedded intelligence in buildings is not new; rather what makes it currently possible are cheap digital sensors, computer power to handle big streams of data, and the development of software specifically developed for on-going operations and maintenance of buildings. Intelligent building systems could in real-time gather data, compare it with past usage, use other forecasts (like climate and pricing), and choose a strategy to optimize something (for example, energy consumption or interior comfort levels). This form of cognitive optimization benefits from the addition of "smart" façade components that respond to the exterior conditions. Early design studies for buildings can test the effectiveness of these systems before they are actually installed both by physical and digital models. Models that respond to environmental sensors (such as temperature or lighting levels) have been successfully built. However, almost always they are created in general 3D design software. This paper demonstrates, with very simple case studies, the linking of an environment sensor (specifically a light level sensor) by using building information modeling (BIM) software, incredibly common in the architecture and construction professions, using Dynamo and the Revit API as interfaces.

2. INTELLIGENT BUILDINGS

Designers have sought both passive methods and active systems to have their buildings respond to changing environmental conditions. Many historic examples exist that show the use of passive methods using natural features to manage daylight, as evidenced at Mesa Verde, or mitigating seasonal heat variation by the use of shade and thermal mass in the southwest United States at Acoma (1).

Active systems have also been incorporated into structures, ranging from those that require direct occupant participation to those that are completely automated building systems with embedded computation. This has evolved into the concept of intelligent buildings and intelligent building facades. Built examples include the automated brise-soleil of the Institut du Monde Arabe (1987), the Commerzbank Headquarters (1997) ventilation and shading strategies, and the automatically adjustable windows of the San Francisco Federal Building (2007). The main design focus for many recent intelligent buildings has been to address a specific environmental condition, whereas the study of environment-human-space interaction has also been studied at a much smaller scale, such as dECOi's Hyposurface, Michael Fox's Bubbles interactive pneumatic environment project, ABI and Zahner's Tessellate (2).

Despite past problems in actuating full size components, the concept of kinetic architecture is flourishing as an exercise in combining aesthetics with energy conservation practices; including daylight harvesting and solar heat gain avoidance (3). This requires what Fox and Kemp refer to as "environmental cognizance", the ability to not only measure values such as temperature, humidity, sun location, solar radiation, rain, etc., but to also have a building be aware of these conditions and respond to those inputs. Intelligent building facades respond to climatic data through a system of sensors and actuators; the controlled response could change the configuration of the space, its services, and climatic needs (3) (4). The interaction of occupant and systems depends on immediate feedback response and is considered the "essential of interaction" (5). However, there is a lack of information, applications, and methods that support the design of an intelligent building and intelligent building envelope (6) (3). Designed properly, an intelligent building should also able to dynamically adapt in changeable environments (7) and be able to address the different environmental conditions over its lifetime due to seasonal and even climatic change.

Currently, the term intelligent building ("smart building") refers to buildings where advanced smart building technology has been installed, usually through the use of building management systems (BMS), which are sometimes connected to a central cloud-based platform. They employ a system of sensors and actuators that modify artificial responsive lighting, daylight controllers, sun controllers, renewable energy systems, automatic ventilation, all through the BMS, which automatically controls the environmental controllers using knowledge-based algorithms and environmental data for indoor and outdoor conditions (8) (9) (10). The intelligent building façade itself acts as the interface between the inside and the outside, readjusting itself based on performance conditions, responding to the interaction of humans and the environment (3).

For example, a "smart building" could monitor occupant behaviour and control building response (in the form of turning off lights or moving adjustable window shades) towards the goal of lessening energy consumption. This is quite common at a low level of intelligence in the use of motion detectors and light sensors to automatically control electrical lighting. Data analytics software provides algorithms for optimizing the building's response. The US GSA (United States General Services Agency, the federal organization in charge of standards for all federal facilities, including sustainable design, construction services, and project management), is implementing a smart building initiative (www.gsa.gov).

3. SIMULATION OF ENVIRONMENTAL CONTROLS

In order to predict the future performance of building components, it is useful to initially create a digital 3D model and connect it to sensors to see if the responses of the components are appropriate. Visual scripting tools are one method for bridging the hardware/software gap between sensors and 3D modeling software. Some designers are effectively and enthusiastically using scripting tools already to generate parametric, form-based solutions for buildings. At NBBJ, architects used Rhino and Grasshopper for the design of the Hangzhou Tennis Center. By defining the geometry of the structure in a visual scripting language, the architects could mathematically describe numerous formal variations in response to various tolerances (11). ThorntonTomasetti used a Rhino/Grasshopper model for the Basrah 30k soccer stadium and automated the Rhino to SAP translation for structural analysis and then sent the final model to Revit using a series of in-house tools (12). Many comparable examples exist in the fields of architecture, construction, and academia.

There is no doubt that visual scripting interfaces have become an important tool for the design and construction of buildings. A current popular program used in conjunction with Rhino is Grasshopper. The preference for Grasshopper could be attributed to the following: its direct relationship to Rhino, a popular 3D modeling program; a substantial group of independent developers who provide support and new components; its

availability free of charge; a lack of comparable features with the same ease of use; and the development of components such as DIVA used for performance evaluations (13). DIVA is an excellent bridge between the 3D modeling done in Rhino / Grasshopper and energy (EnergyPro) and daylight calculations (Radiance/Daysim) (14). In one example, moveable light shelf angles were optimized for daylight availability. The designer used DIVA and Daysim for daylighting calculations with Rhino and Grasshopper and Galapagos (a genetic algorithm for optimization). That study demonstrated that a kinetic façade system can contribute to better illumination levels in a space through daylight harvesting, in turn reducing the reliance upon artificial lighting systems and saving energy energy (15). It provides an accessible method of using environmental performance simulation via a scripting interface. Other components in Grasshopper provide database capabilities (Slingshot!), live physics engine (Kangaroo), form finding and structural analysis (Geometry Gym), connection to Arduino and other input/output devices (Firefly), and many others.

However, Rhino itself is not a building information model (BIM) tool, and as a 3D surface modeling program, it is not directly interoperable with BIM software. Researchers are developing scripting or graphical algorithm editors to create a connection between parametric modeling and environmental simulation. One work in progress is the development of a parametric modeling based method for evaluating façade configurations for hot and humid climates (16). "Daren Thomas of the Professur für Gebäudetechnik, Institut für Hochbautechnik at the technical university ETH Zürich has published a Python Shell for Revit. It was implemented using IronPython and is used to automate the running of daily tests of a building energy analysis package" (17).

As building information models are often used in the architecture and construction industries, having a visual scripting interface for commonly used BIM software programs, such as Autodesk Revit, would be incredibly useful. Dynamo is one scripting interface that is under development.

4. DYNAMO: A VISUAL SCRIPTING INTERFACE FOR THE REVIT API

The Autodesk Revit API (Application Programming Interface) allows users to add to the features of the software and create custom tools and plugins. Dynamo is under development by users as a plugin to Revit using the Revit API and built using the Windows Presentation Framework. Dynamo's look and feel is influenced by a number of visual programming interfaces that have come before including MaxMSP, the Maya Hypergraph, and LEGO MINDSTORMS NXT, which is based on National Instrument's LabVIEW (18). As a parametric modeling engine, Dynamo takes its inspiration from Bentley's Generative Components and McNeel's Grasshopper for Rhino. It is designed to extend Revit's parametric modeling capabilities by adding a level of associativity that does not exist in the off-the-shelf application including driving parameters based on external inputs, such as sensors or by data taken from an analysis. One can map the appropriate parameters and dynamically change each value with a value derived from the input source.

Although not nearly as the stage of development of Grasshopper, Dynamo has been used for several applications, one of which is the connection between the Arduino board, a light sensor, and a building information model that this paper discusses. As a general purpose interface for Revit and Vasari, it has been used in other applications, for example: creating making Revit views and sheet layouts, this has a potential use for fabrication of architecture components (19); creating virtual automatic shading devices (20); extracting solar radiation values and driving Revit parametric geometry by manipulating a physical slider with the use of the Arduino board (21).

The elements with which users interact in Dynamo are referred to as "nodes". Each node can have a number of "ports", which enable communication between nodes along "connectors". Ports can only be connected to other ports whose output type matches the port's input type, or to any port whose output type is further up the inheritance hierarchy of the port's input type. Together these connected elements create the "workflow". The base class for Dynamo, from which all elements inherit, is dynElement. As an abstract class, dynElement provides the framework on which all elements are built. Included in this class are the three methods that comprise the "build loop" of Dynamo: destroy, draw, and update. The destroy method is used to clean up elements or objects created and owned by the node. The draw method is used to generate Revit elements or objects based on the input connections. And the update method is used to instruct "downstream" nodes to execute their own build loops. Successive downstream nodes are evaluated until either an exception is raised on a node or there are no more nodes to process. A user can implement custom functionality in the build loop by overriding dynElement's destroy, build, or update methods (18).

Nodes are classified as "transactional" or "non-transactional". Transactional nodes open a database transaction during their build phase, and all elements created during that phase are committed to the database; this essentially means that transactional actions allow the building database to be updated. Non-transactional nodes do not open transactions as they do not need to change the Revit database. An example of a transactional node would be one that creates or edits a group of family instances. A non-transactional node, by comparison, could be one that interacts with serial data such as an Arduino board, allows the input of a numeric value, or provides an answer to a query about the database (for example, what is the total square footage of floors) without changing anything in the database. If a node throws an exception or error during processing, the database transaction is cancelled, and the database is rolled back to its prior state (18).

5. ARDUINO MICROPROCESSOR

Arduino (http://arduino.cc) is a single chip microcomputer that executes programs created in the Processing programming language. One can use it to as a software interface, control system for robots, data recorder, kinetic responsive art installations, and other applications. The Arduino is popular with hobbyist for two main reasons: it needs minimal knowledge of programming or electronics to use, and there exist a variety of sensors (environmental and others) that are easily attached to it including those for motion, temperature, humidity, lighting levels, air quality. Servos can also be controlled by the Arduino. Scientists have used the Arduino board in diverse applications such as an LED simulator system (22), and participatory sensoring of air quality (23) and used with the open-source Cosm platform for transmitting real time sensor data through Internet feeds (24).

6. CASE STUDY METHODOLOGY AND RESULTS

Eight case studies (seven were successful) were accomplished to demonstrate the feasibility of connecting environmental sensors to control a building information model (BIM) and established that the process could go in both directions, from real models to virtual models and from digital models to physical models. The eight case studies were as follows:

1. Arduino photoresistor, Firefly, Grasshopper, Rhino (simple model)
2. Arduino photoresistor, Dynamo, Revit (simple model)
3. Arduino photoresistor, Dynamo, Revit (panel, louver, overhang)
4. Arduino photoresistor, Dynamo, Revit (façade component)
5. Arduino photoresistor, Dynamo, physical model (façade component)
6. Revit 3D model, Dynamo, Arduino servo, physical model (façade component)
7. Arduino photoresistor, Revit dll, Revit (façade component)
8. Revit 3D model, Revit dll, Arduino servo, physical model (façade component)

Case study 1 was the base case to demonstrate a simple link using Rhino (a non-BIM 3D modeling program) and the sensor. Rhino with Firefly and Grasshopper is a common solution for this type of connection. Case study 2 duplicated this result.

In case studies 1, 2, 3, 4, and 7, the lighting levels registered by the photoresistors caused the digital model to change. This was the initial scope of this research project – real sensor data caused a building information model to respond interactively.

In case study 5, the lighting level caused a physical model to move. Although not part of the original study, it seemed useful that if a real sensor could activate virtual models, a physical model should not be difficult. This lead to the next two case studies, 6 and 8, where the Revit model was edited, thus causing the physical model to update. The intent here was to show that data in the virtual world (for example, the sun moving to different dates and times) could be used to actuate a physical model. Bi-directionality between the virtual and real was easily achievable.

Case studies 2 – 6 used Dynamo as the link between the sensor's output and the models (both physical and digital). However, at that time, the bi-directionality link of Dynamo was not working properly. The last set of case studies (case studies 7 and 8) by-passed the use of Dynamo, and instead the Revit API was used to establish the passing of data between the Arduino (photoresistor and servor) and the models.

6.1. Arduino Photoresistor, Firefly, Grasshopper, Rhino (simple model)

A simple model was created in Rhino and Grasshopper that reacted to values output by a photoresistor on an Arduino board. Firefly was the interface between Grasshopper and the Arduino.

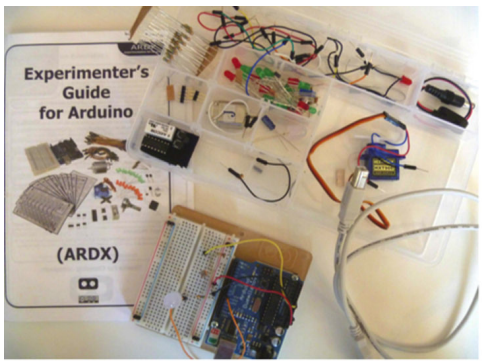
This base case consisted of three elements: a wall surface, a window opening, and a window shade that could move. Conceptually, the window shade dynamically responds to the presence of a sun by moving horizontally above the window opening to effectively shade the interior space. In reality, the initial models consisted of a box moving along the face of another box. The distance that the shading device moves is controlled by input from a photoresistor on the Arduino board (Figure 1).

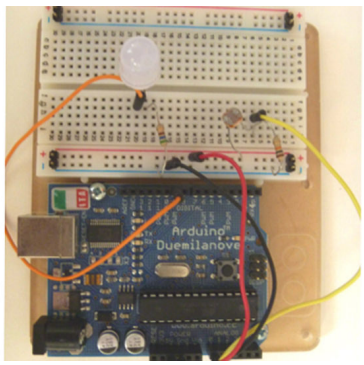
The first model was constructed in Grasshopper with the 3D model appearing simultaneously in the Rhino view window (Figure 2 – left side). The inter-relationship between the window shade and wall is explicitly defined and can be manipulated given a change to any particular dimension, specifically the amount of distance that the shading device would move (the parameter, "offset"). This offset distance is related to the measured light level value. To procure this input data, a Processing sketch is uploaded to Arduino to register analog values from a photoresistor and report them to Firefly. The Firefly component for Grasshopper receives these values and through simple mathematical operations in Grasshopper, translates them into displacement distances for the window shade. The response of the 3D Rhino model is practically real-time. As a user moves his hand off of the photoresistor and the lighting levels increase, the window shade moves. The next step was to do this same exercise using Revit and Dynamo.

6.2. Arduino Photoresistor, Dynamo, Revit (simple model)

The same simple model from the first case study was re-created in Autodesk Revit. Dynamo was used as the link between Revit and the Arduino board. The photoresistor on the Arduino board output values that controlled parameters in Revit.

Parametric objects in Revit are called "families". They have "instance" parameters that refer to an individual object and






```
text/*
output from analogue pin 0 (light sensor)
  used photo-resister example for base of this sketch
*/
//PhotoResistor Pin
int lightPin = 0;
int delayTime =1000;
void setup()
{
 Serial.begin(9600);
}
void loop()
{
 int lightLevel = analogRead(lightPin);
 //read the light level
 lightLevel = map(lightLevel, 0, 900, 0, 255);
 //make sure the values are between 0 and 255
 lightLevel = constrain(lightLevel, 0, 255);
 Serial.write(lightLevel);
 // delay(delayTime);
}
```

Figure 1. Arduino kit (left) Completed circuit board (without USB cable; the LED was only used for testing purposes (middle); Processing code (right).

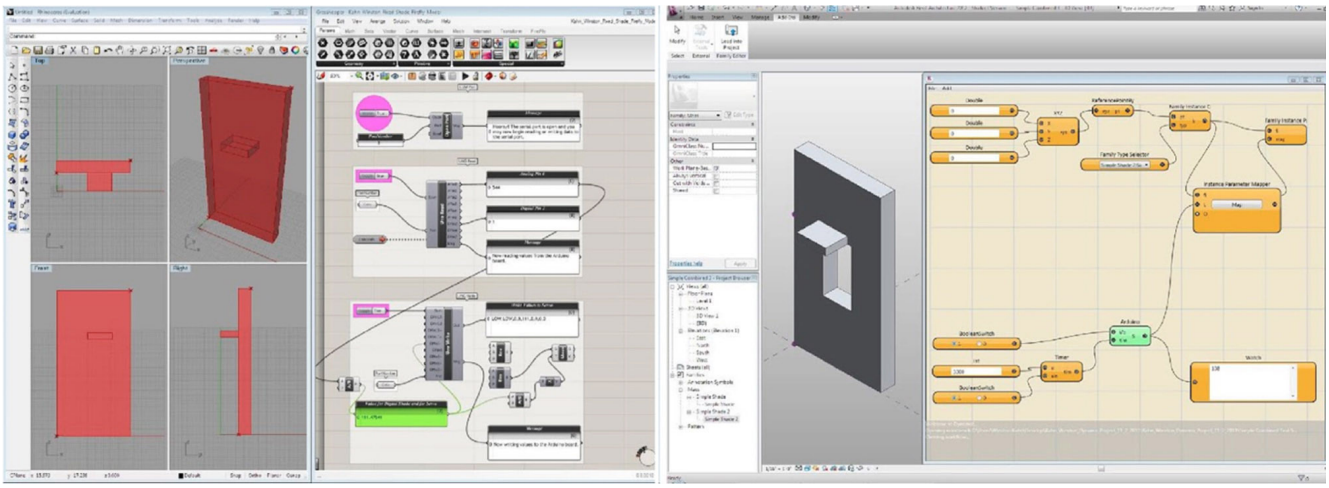


Figure 2. Rhino, Grasshopper, Firefly (left); Revit, Dynamo (right) (images by Winston Kahn.

"type parameters" that refer to a category of objects. Users can add new parameters to "loadable families" and to "conceptual masses". To modify forms in the Revit conceptual massing modeler, one can directly manipulate the model as in Rhino or change parameter values. Objects were adjusted by Dynamo through their parameters.

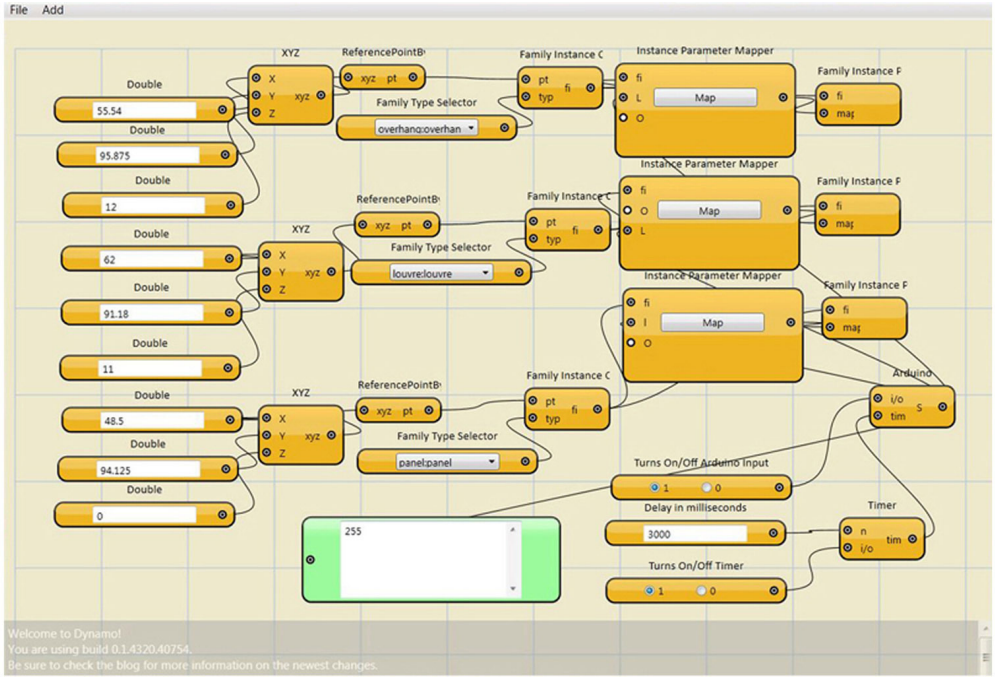
The model created in the Revit conceptual modeler visually matched the first Rhino model. First the window shade was created. It was given two instance parameters: Light Level is a value from 0 to 255 that is received from the Arduino board; Offset Shade sets the distance of the shading device from the edge of the window based on Light Level. In a second file, the wall was created. The shading device family was inserted into the wall model. It was verified that the shading device was on the correct location on the wall, that it behaved properly based on changing the parameters values manually (a bit of tweaking took place here to convert the 0-255 values to reasonable distances), and that Arduino was providing data that Dynamo was receiving correctly. Once these nodes were connected, the shade was moved by Dynamo in Revit directly in response to the Arduino light level readings, although more slowly than in the first case study (Figure 2 – right side).

6.3. Arduino Photoresistor, Dynamo, Revit (panel, louver, overhang)

A more complex 3D model was created in Revit that better portrayed the potential of controlling a 3D architecture model from the output of a photoresistor. Unchanged from previous case study, the photoresistor on the Arduino board output values that changed the parameters of components in Revit with Dynamo as the connective software.

The panel, louvers, and overhang were interactively changed based on input from the photoresistor. As the user's hand passed over the photoresistor, the value output changed from almost 0 to 255. Slowly, the Revit model updated the size of the holes in the dynamic panel, the rotation of the louvers, and the length of the overhang on the house (Figures 3 and 4).

The data on the left and upper parts of the workbench are the values to position the dynamic panel, a louvered panel, and an overhang on the house.



The lower right corner contains the nodes for connecting to the output of the Arduino board and light sensor and using that information to change parameter values of the families.

Figure 3. Dynamo input screen called a "workbench".

Figure 4. 3D model in Revit: light level is 255 (left). 3D model in Revit: light level is 75 (right) (the original house was modeled in Revit by Andrea Martinez).

6.4. Arduino Photoresistor, Dynamo, Revit (façade component)

Similar to the previous case study, the building information model responded to the changes in the photoresistor's value (Figure 5).

6.5. Arduino Photoresistor, Dynamo, physical model (façade component)

In this case study, both the physical model responded to the changes in the photoresistor's value. As the light level value approaches 255, all three vertices approach a maximum extension in the physical and digital models, likewise, when the value reading is at a lower threshold, only one or two or three of the actuators or vertices are raised (Figure 6).

6.6. Revit 3D model, Dynamo, Arduino servo, physical model (façade component)

The setup is ready for this case study. However, as mentioned earlier, at the time the case studies were finished, there was no supported functionality in Dynamo to output the Revit parameter values to the Arduino board.

6.7. Arduino Photoresistor Revit dll, Revit (façade component)

Dynamo was successful for mediating between the photoresistor and Revit model, but not from the Revit model to a physical model and at a very slow speed. Another method was tried using a custom dll plugin written in the Revit API as the connection between the photoresistor on the Arduino board

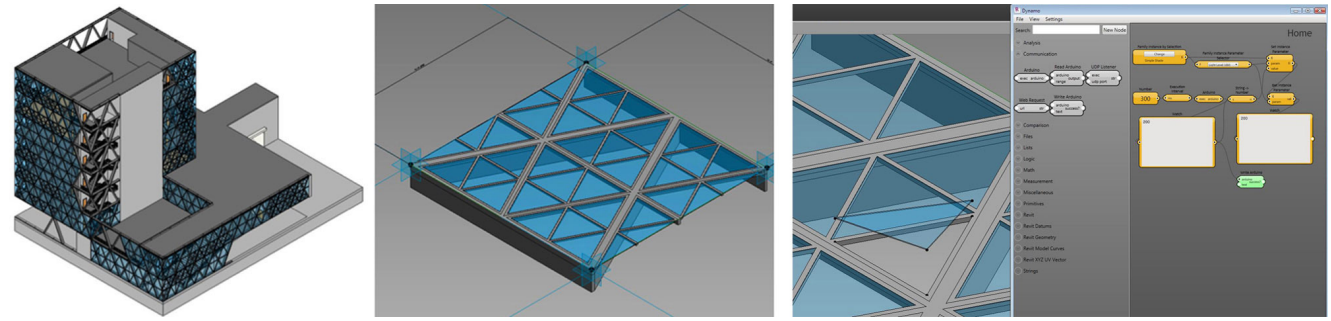


Figure 5. Revit model of building, façade component, and façade component in Dynamo (images by Winston Kahn).

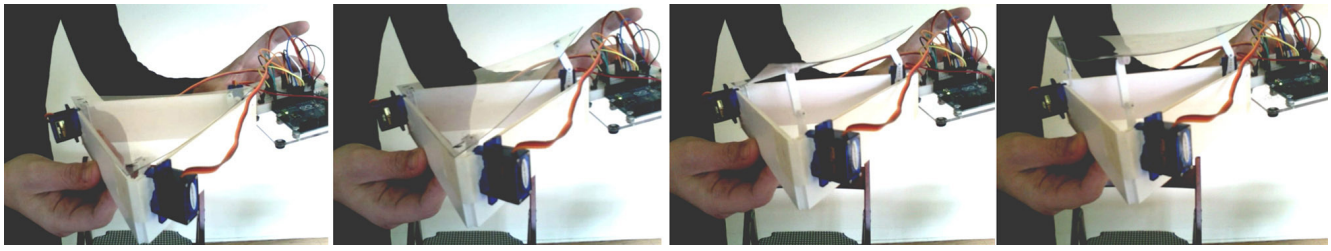


Figure 6. Physical model of the façade component with zero, one, two, and three corners raised (images by Winston Kahn).

and the Revit model. This involved an extra step that involved writing out the light level values from the Arduino to a text file that was input to Revit. In addition, only one value at a time could be sent for each running of the Revit dll (Figure 7). More clever coding in the future would directly link the output to Revit and provide for a serial stream of light level values.

6.8. Revit 3D model, Revit dll, Arduino servo, physical model (façade component)

In case study 8, a link from the Revit 3D model to a physical model was accomplished. Values can be changed in an angle parameter in Revit, and the physical model responds (Figure 8).

An unfinished feature is to access the values in the sun component in Revit. Then as the sun "moves" in Revit, it would jointly change the shading device angle both in Revit and in the physical model.

7. DISCUSSION AND CONCLUSION

Although the examples shown are simple in construction, they prove that both workflows (the use of Dynamo as a visual scripting language and use of the Revit API to create a dll plugin) were successful in using a light sensor to drive parameters in a building information model and changing parameters in a 3D model to provide input for servos to move a physical model. Some difficulties were encountered, and the entire topic area of optimization was not explored in this case study.

7.1. Difficulties

The interaction time in Dynamo/Revit is vastly slower than Grasshopper/Rhino. There were noticeable delays when the users moved their hands over the photoresistor. A primary challenge to overcome in the development of Dynamo is that

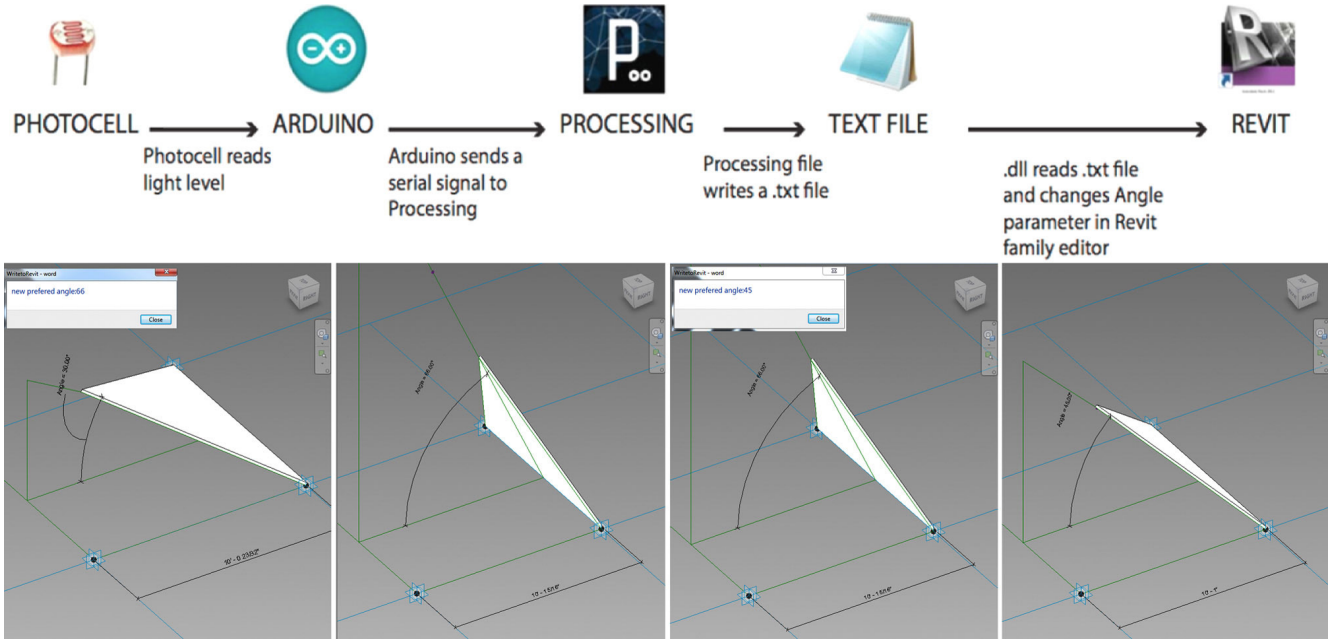


Figure 7. Diagram of process (top). Sequence of running the dll in the Revit family editor (bottom): Revit acquiring the value from the photoresistor and applying it for 66 degrees and then 45 degrees. The angle parameter updates the geometry (images by Augustine Liu).

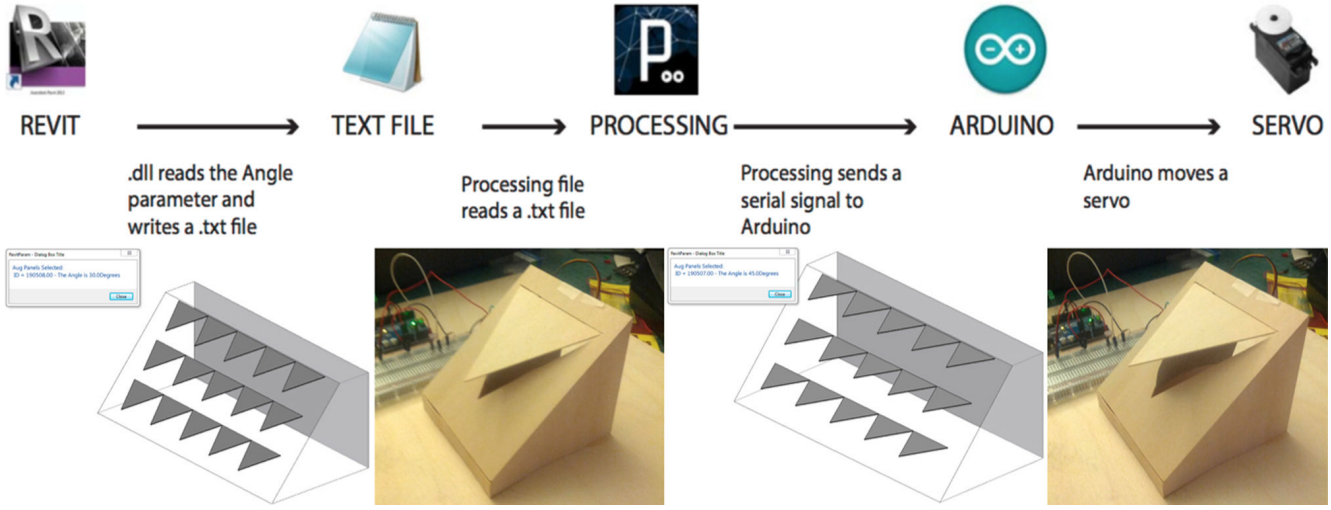


Figure 8. Diagram of process (top). Parameter is 30 degrees in digital/physical models (bottom – left). Parameter is 45 degrees in digital/physical models (bottom – right) (images by Augustine Liu).

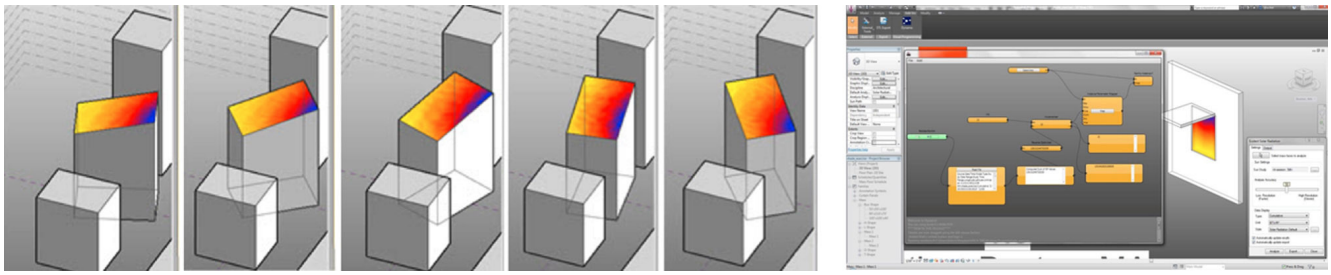


Figure 9. Left - Progression of images showing radiation analysis until local maximum was reached (image courtesy of Matt Jezyk, Autodesk). http://wikihelp.autodesk.com/Vasari/enu/Community/Work_in_Progress/Dynamo_for_Vasari (last accessed 1-25-12). Right - Progression of images showing radiation analysis until a local maximum was reached (image courtesy of Tyler Tucker).

of the speed of interactions with the database. Unlike Grasshopper, Dynamo has no concept of "volatile" geometry, that is, geometry that is visualized but does not get committed to the database until the user "bakes" it. For Grasshopper this provides a benefit in terms of the speed with which a workflow can be evaluated. In Dynamo, the evaluation speed of the workflow will always be tied to the speed with which its database transactions can be carried out. In addition, all geometry derived from a Dynamo workflow is "live", having been already committed, which creates an inherent fragility as a user can then manipulate, or in the worst case, delete geometry directly in the Revit interface, subsequently breaking the workflow (18).

7.2. Optimization

The Dynamo and Revit plugin models demonstrated a method of dynamically re-positioning a design element in response to an environmental stimulus. The method was not predicated upon approaching optimization with regard to solar radiation values. However, a starting point for conceptualizing a more complex shading device could begin with an analytic process was previously demonstrated by a team at Perkins + Will. They were able to export model geometry created in Revit, test it against solar radiation data from Ecotect, and ultimately import that Excel spreadsheet data back into Revit -- effectively translating said values into instance parameter changes (25).

Many projects have used Galapagos (an evolutionary optimization module) with Rhino, Grasshopper, and DIVA (simulation) for optimization of daylighting and energy savings. One recent example optimized window sizes for different climate zones and orientations by balancing daylight capture with increases in summer cooling loads (26). Many fewer examples exist for Revit as a module similar to Galapagos does not currently exist for Dynamo. However, other methods have been used to find optimal configurations. Dynamo has been used with Vasari/Revit to create an interactive feedback loop where the amount of solar radiation on the roof of a massing model is optimized by rotating the building (Figure 9 – left side).

The Dynamo setup in Figure 9 – bottom followed a similar sequence as the solar radiation project shown in Figure 9 – top. The *Reverse Optimizer* node replaced the *Optimizer* node and the *Incrementer* node influenced the increment parameter of the shade. The shade depth continued to increase in size until the radiation value on the window no longer decreased, thus optimizing the shade length.

7.3. Future Work

Although there is much to be done, the author foresees two major next steps for the advancement of these case studies. The first is focused on the interaction between environmental sensors and optimization algorithms to predict the changes that should be made to an interactive façade. An interesting subset of this would be to source data from real time weather feeds on the Internet such as Xively (formerly Cosm and before that Pachube). Another step is to construct a full size mock-up to see if it performs like the prototype hybrid virtual-real models. One known challenge to this is the up-scaling of power required to actuate larger servo drives, which in turn are moving significantly heavier construction materials. Micro servo-motors are powered through the Arduino board and operate with a maximum of five volts. In both cases, digital models would first be used to test the response of the systems.

The long-term goal is to create cognizant building facades that optimize their performance based on current, historical, and weather predictions in an effort to reduce energy consumption. Creating virtual / scale-model prototypes is a first step towards achieving that goal. This research concludes that it is possible to connect environment sensors to building information models using Arduino and Dynamo or the Revit API.

ACKNOWLEDGMENTS

Ian Keough, creator of Dynamo; Matt Jezyk, Autodesk; student researchers: Winston Kahn (case studies 1, 2, 4, 5, and 6), Augustine Liu (case studies 7 and 8), Tyler Tucker; Yara Masri, a PhD student studying intelligent building envelopes. An earlier version of this paper was presented at BESS-SB13 CALIFORNIA (Building Enclosure Sustainability Symposium / Sustainable Buildings 2013) conference, Pomona, CA, June 2013.

REFERENCES

(1) Knowles, R. L. (1981). *Sun Rhythm Form*. pp. 11-13, 158. Cambridge: MIT Press
(2) Masri, Y. (2013). Intelligent building envelopes: a design methodology (PhD dissertation). L.A.: Univerity of Southern California.
(3) Fox, M., Kemp, M. (2009). *Interactive Architecture*. pp. 109-117. Princeton: Princeton Architectural Press.
(4) Sterk, T. (2005). Building upon Negroponte: a hybridized model of control suitable for responsive architecture. *Automation in Construction* 14(2): 225-232, doi: http://dx.doi.org/10.1016/j.autcon.2004.07.003.

(5) Haque, U. (2006). Architecture, interaction, systems. *AU: Arquitetura & Urbanismo* (149): 1-5. http://www.haque.co.uk/papers/ArchInterSys.pdf.
(6) Yang, J., Peng, H. (2001). Decision support to the application of intelligent building technologies. *Renewable Energy*, 22(1–3): 67-77, doi: http://dx.doi.org/10.1016/S0960-1481(00)00085-9.
(7) Compagno, A. (2002). *Intelligente Glasfassaden/Intelligent Glass Façades: Material, Anwendung, Gestaltung/Material, Practice, Design*. Basel: Birkhäuser.
(8) Wigginton, M., Harris, J. (2002). *Intelligent Skins*. Oxford: Butterworth-Heinemann.
(9) Selkowitz-Stephen, E. (2001). Integrating advanced façades into high performance buildings. En *Glass Processing Days: 7th International Conference on Architectural and Automotive Glass*. Tampere, Finland.
(10) Smith, S. (2002). Intelligent buildings. En Best, R., Valence, G. (Eds.) *Design and Construction: Building in Value,* (pp. 36-57). Oxford: Butterworth-Heinemann.
(11) Miller, N. (2011). The Hangzhou tennis center: a case study in integrated parametric design. En *ACADIA Regional 2011 Conference: Parametricism (SPC)*, (pp. 141- 148).
(12) Burns, J. (2011). Stretching BIM from design to construction. En *USC Symposium, EXTREME BIM 2011*. L.A.: University of Southern California.
(13) Reinhart, C. (2013). DIVA home page. http://diva4rhino.com/profile/Tito.
(14) Lagios, K., Niemasz, J., Reinhart, C. (2010). Animated building performance simulation (ABPS) - linking Rhinoceros/Grasshopper with Radiance/Daysim. En *SimBuild 2010 Conference proceedings,* (pp. 321-327). New York.
(15) El Sheikh, M., Kensek, K. (2011). Intelligent skins: Daylight harvesting through dynamic light-deflection in office spaces. En *ARCC 2011 Conference proceedings*, (pp. 293-304). Detroit.
(16) Velasco, R., Robles, D. (2011). Eco-envolventes: A parametric design approach to generate and evaluate façade configurations for hot and humid climates. En *eCAADe 29 th Conference Proceedings*, (pp. 539-548). Ljubljana: University of Ljubljana.
(17) Tammik, J. (2011). http://thebuildingcoder.typepad.com/blog/2009/12/revit-python-shell.html. Also visit http://www.youtube.com/watch?v=IuVwn-U91Hc for a demonstration of it.
(18) Keough, I. (2011). Dynamo: designing a visual scripting interface for the Revit API (notes). Also see https://github.com/ikeough/Dynamo/wiki for more information about coding in Dynamo
(19) Kron, Z. (2013). Buildz - practical notes for making impractical things - making Revit views and sheet layouts in Dynamo. http://buildz.blogspot.com.
(20) Scheer, D. (2013). Fun with Dynamo for BPA - Automatic shading design. http://autodesk.typepad.com/bpa/2013/08/more-fun-with-dynamo-for-bpa-automatic-shading-design.html.
(21) Jezyk, M. (2013). Revit Dynamo Visual Programming connected to Arduino, http://www.youtube.com/watch?v=VidwY8Ki4ZQ.
(22) Teikari, P., Najjar, R., Malkki, H., Knoblauch, K., Dumortier, D., Gronfier, C., Cooper, H. (2012). An inexpensive Arduino-based LED stimulator system for vision research. *Journal of Neuroscience Methods,* 211(2): 227-236, doi: http://dx.doi.org/10.1016/j.jneumeth.2012.09.012.
(23) Mendez, D., Labradorb, M., Ramachandranc, K. (2013). Data interpolation for participatory sensing systems. *Pervasive and Mobile Computing*, 9(1): 132-148, doi: http://dx.doi.org/10.1016/j.pmcj.2012.11.001.
(24) Papageorgas, P., Piromalis, D., Antonakoglou, K., Vokas, G., Tseles, D., Arvanitis, K. (2013). Smart solar panels: in-situ monitoring of photovoltaic panels based on wired and wireless sensor networks. *Energy Procedia*, 36: 535-545, doi: http://dx.doi.org/10.1016/j.egypro.2013.07.062.
(25) Aksamija, A., Guttman, M., Rangarajan, H.P., Meador, T. (2011). Parametric control of BIM elements for sustainable design in Revit: linking design and analytical software applications through customization. *Perkins + Will Research Journal*, vol. 03-01: 32-45
(26) Wu, G., Kensek, K., Schiler, M. (2012). Studies in Preliminary Design of Fenestration: Balancing daylight harvesting and energy consumption. En *PLEA 2012 Conference proceedings*. Lima.

* * *