The Smart Commons:
An Experiment in Sensor-Based Space Assessment of Learning Environments

*Jonathan Bradley, Patrick Tomlin, and Brian Mathews**

A kitchen refrigerator. What could such a mundane object tell us about ourselves? About its environment? Not much on its own, perhaps. But what if it were augmented with data-collecting technology that could transmit status reports to your smart phone? This refrigerator might then be able to compile and send a real-time list of the items stored within it to your tablet or smartphone; make orders with Amazon to replace consumed food; provide a recipe based on already-stocked ingredients; track expiration dates; or adjust its internal temperature to maintain the maximum freshness of its contents. It would be a part of the increasingly ubiquitous network of smart objects called the Internet of Things (IoT).

Refrigerators, lamps, headphones, trash cans: these and a growing number of everyday devices are embedded with sensors, actuators, and software capable of processing, storing, and wirelessly transmitting data about themselves over a network using internet protocol. Businesses employ IoT devices to lower operating costs by remotely monitoring wear and tear on manufacturing equipment, track sales and deliveries, and relay information about customer interaction with inventory in stores. The health industry utilizes IoT technology to monitor patients' blood pressure, heart rate, and implanted devices such as pacemakers and hearing aids. Farmers are even attaching sensors to their crops, allowing them to monitor and aggregate data about weather and soil conditions. As Jeremy Rifkin writes, "The IoT embeds the built environment and the natural environment in a coherent operating network, allowing every human being and everything to communicate with one another in searching out synergies and facilitating interconnections."[1] In other words, nearly every aspect of life has the capability to be enhanced by the Internet of Things. And that interconnection is already occurring: forecasts suggest that 34 billion devices, from wearable objects like Fitbits and Apple watches to digital signs, will be connected to the internet by 2020, up from 10 billion in 2015.[2] (OCLC, 2015).

The advent of the Internet of Things holds potential for the library as well. From gate counts to checkout statistics, libraries have long invested in data collection as a means of creating and improving services. As more libraries shift their attention to assessing user experience and gauging the impact of their spaces, user studies employing ethnographic strategies such as observations and interviews have taken on greater prominence. Survey instruments such as LibQual+ provide an entry point into the perceptions of users. Yet what is missing from all of these approaches, whether quantitative or qualitative in nature, are real-time results and a holistic picture of the library. How do users interact with the library? How can we customize library spaces and services to provide an optimal user experience? Utilizing sensor-based technology might begin to provide an answer.

During the Fall of 2016 and Spring of 2017, we set out to test the potential application of IoT technology to better understand our users' interactions with the Library's learning commons. Beginning with tracking the movement of commons furniture—following a "day in the life" of a single whiteboard, for example—we hoped

* *Jonathan Bradley is Web Learning Environments Application Developer, Virginia Tech Libraries; Patrick Tomlin is Associate Director of Learning Environments, Virginia Tech Libraries; Brian Mathews is Associate Dean for Learning, Virginia Tech Libraries.*

to illuminate patterns of student work, examine the density of particular work areas, and create better designed, more effective learning spaces and user experiences. Through the use of Bluetooth and Wi-Fi-enabled development boards such as the Arduino family and the Raspberry Pi, numerous sensors were monitored, their data fed back to a locally hosted server. Using accelerometers, motion detectors, force sensors, and Bluetooth beacons, we created a system for monitoring where and when library commons furniture, equipment, and study rooms were occupied or moved, and how often students interacted with them. All data was pushed to a local server where it was collected for analysis and visualization.

The ecosystem built around the IoT offers curious users with ample opportunity to experiment with projects; most of the sensors and controllers are cheap, costing a couple of dollars up to $35 for a full microcomputer like the Raspberry Pi, and aside from some particularly sensitive modules, the devices are hard to permanently disable due to accident. The IoT ecosystem has also given birth to a huge community of enthusiasts offering support and code to get a prototype up and running, and much of this ecosystem relies on open-source hardware and software. With a low up-front cost and good support, the IoT can offer safe and effective exploration of new ways to collect assessment data.

The following sections break down and describe the various components of the Smart Commons project.

Technology

At the onset of this project, the choices behind which hardware and software would be used followed a few simple guidelines. First, the hardware and software should be readily available and as cheap as possible. As a project like this scales, the cost can grow tremendously, and while this might be fine for a large research institute, it does not translate well to smaller universities or colleges that may hope to replicate such a system for their own purposes but do not have the funding to purchase expensive equipment. Having the sensors and other hardware be cost effective also allows for low-stakes testing and proof-of-concept needed to get buy-in for stakeholders who may not initially see the value of such a project.

The cost of a standard sensor module like the ones we used for the chairs and whiteboards is roughly $60, which includes all the sensors, controller, buttons, and batteries. This cost could be cut to as low $25 if the user was willing to charge the battery more often and felt as though the accelerometer was not necessary. We found that the $60 build provided a good balance of efficiency and cost. The low cost also allows for easy replacement of a sensor is damaged or stolen. All of the sensors and the board used in the project can be purchased online at retailers such as Adafruit and Allied Electronics, and no custom PCB manufacturing is required, making the barrier to entry low for anyone with an interest and mind for electronics.

The second goal with the hardware was that it would utilize as many open systems as possible. Most everything in one of our sensor modules has full schematics and details online, and the controller board has a good community behind it, which also lowers the barrier to entry since a wealth of troubleshooting documentation already exists for similar projects. This dedication to open systems is in keeping with the culture surrounding libraries, and as an extension, we have open-sourced the code, schematics, and 3D printed designs used in this project.

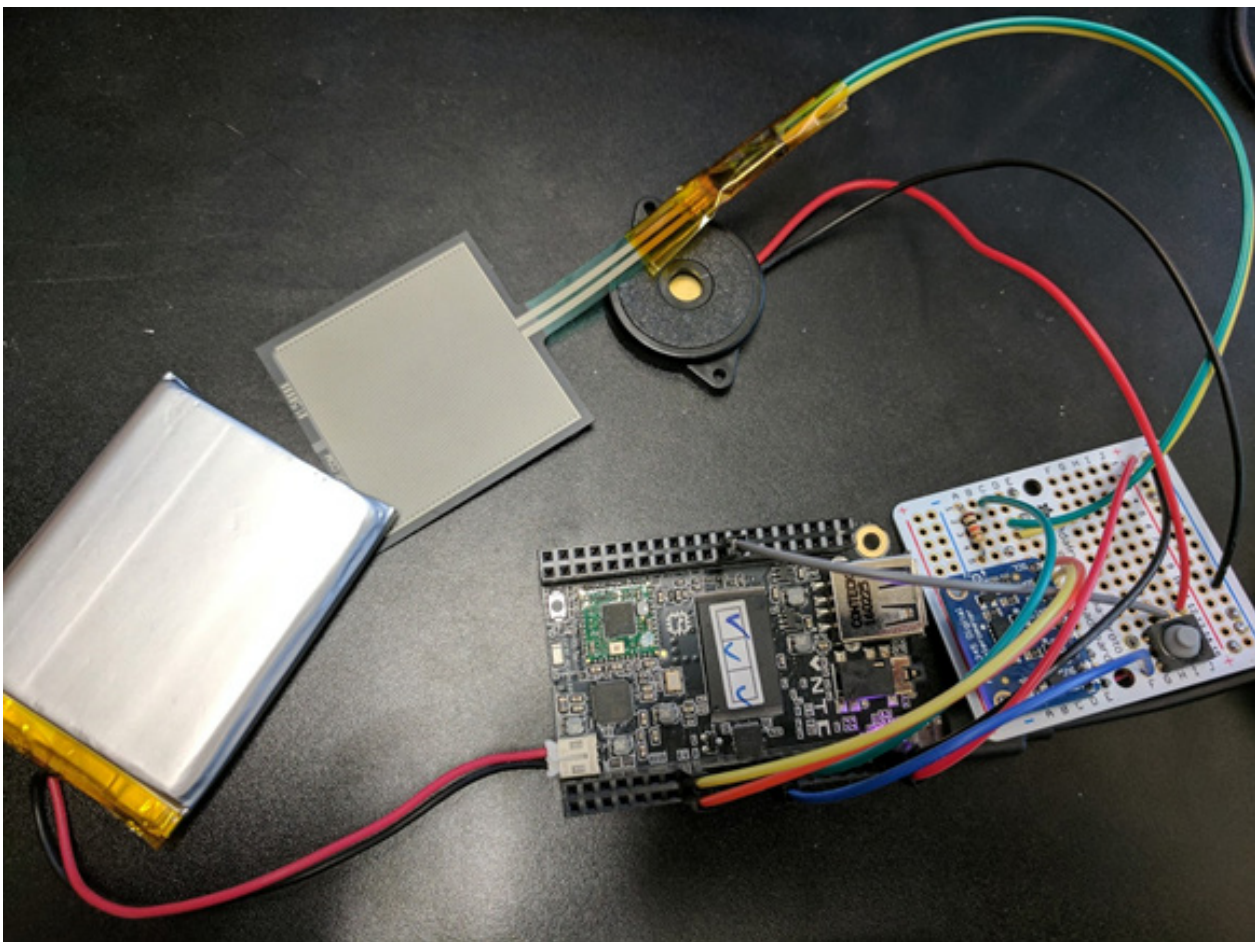
These considerations stretch to the software used as well. The project does not use any proprietary software, only open-source projects available in repositories like Github. The controllers run a robotics package called Johnny-five, which sits on the Node.js engine, meaning all of the coding is done in JavaScript. This decision was deliberate, since JavaScript is the most commonly known programming language and an easy one to learn for beginner who may be interested in a similar project. This decision also allows for easier distribution of the project through npm, the Node Package Manager, and installation can be as easy as running a few predefined commands on the controller.

The Hardware

The Controller

As the brain behind the sensors, the choice of control board has been the most agonized over. The choice has shifted from Raspberry Pis to Arduino-based controllers, before finally settling on the C.H.I.P. from Next Thing Co. The C.H.I.P. was chosen because it offer built in Bluetooth and wireless, something none of the current Arduino boards offered, and cost $9 each, significantly cheaper than the Raspberry Pi 3, which runs $35 each. The C.H.I.P is also has a full ARM processor, unlike the Arduino family which use 8 bit processors, meaning it offers flexibility on what programming languages could be used. In this case, the Node.js environment used in this project would be unable to run on the Arduino family. However, the trade-off is battery life. Many of the Arduino-based controllers can get exponentially better battery life than the C.H.I.P or Raspberry Pi. In the future, we may switch out the C.H.I.P. boards for an ESP32-based controller, which offers better battery life and both Bluetooth and wireless for a similar price as the C.H.I.P. At the time this project began, however, the development modules for the ESP32 had just been released and the very little documentation or even IDEs existed for it yet, but as the platform matures, it might be a viable alternative to the current setup. It would require rewriting the code, but the benefits to battery life might very well justify the change.

FIGURE 1
Completed Smart Commons Module and Controller



Sensors

ACCELEROMETER

The project uses an ADXL345 accelerometer breakout from Adafruit, which costs approximately $17 and is connected via i2c on the C.H.I.P. controller. In the current code, the controller is less interested in which direction the chair or whiteboard is being moved, since its orientation in the space can vary and would require repeated calibration to track accurately, and more concerned with whether or not the object is being moved. The configuration file contains a setting for sensitivity, which defines the threshold for a piece of data being sent, and adjusting this can help get more readings or fewer depending on the amount of data desired. The configuration file also contains a setting for a bounce timer, which is the minimum time between readings. Setting this higher will prevent multiple readings from a single patron "action", so it can be used to decrease the amount of data "noise" you receive from the sensor.

FORCE SENSITIVE RESISTOR

The FSR is used in the chairs to track when patrons sit and stand, and for this project we used the Square Force Sensitive Resistor Interlink from Adafruit, which costs approximately $7. The FSR has a longer cable than the other sensors to accommodate the fact that it is wedged under the seat of the chair. We chose chairs for this project that have cushioned seats and a cavity underneath in which we could insert the FSR. Any chair with a cushion could feasibly work with this module, though some may require more work to get it attached and reading than others. Chairs with hard seats and no cushion are bad candidates, as the only way to get readings would be to have the patrons sit directly on the sensor, a prospect that is likely to result in fewer students using that seat versus another available chair. The FSR has no settings in the configuration file because by default it is attached to a digital pin, meaning the sensor is either triggered or untriggered. For certain setups, the configuration could be adjusted and the sensor connected to an analog pin, which would allow for a threshold to be set that would mark whether a patron was using the chair or not; this could be useful for chairs in which the pressure on the sensor was not that great when someone was sitting in the chair.

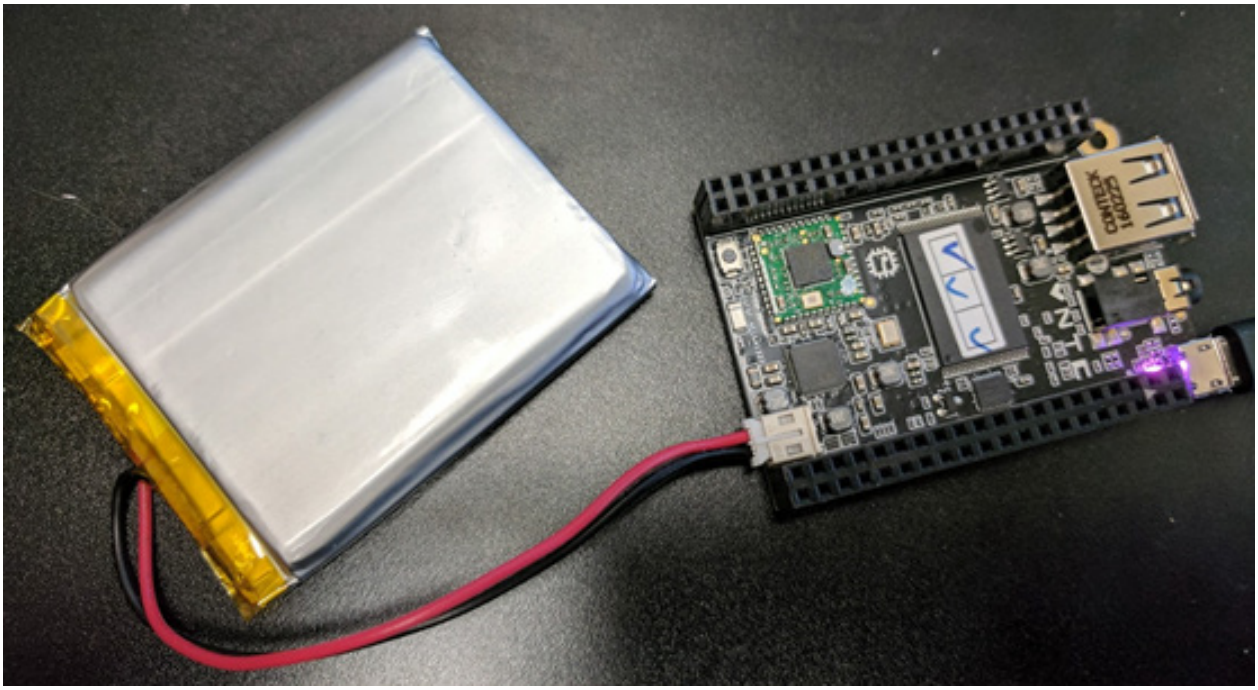
BLUETOOTH BEACON LOCATION

The C.H.I.P. onboard Bluetooth allows the system to do location tracking via Bluetooth Low Energy Beacons. We have the Estimote iBeacons already set up in the library for a self-guided library iPad tour app, so we were able to tap into this existing resource for this project. However, you can purchase beacons if needed for around $15 each that are compatible (iBeacon protocol) with the code provided and attach easily to walls or ceilings. The configuration file contains a number of settings that are needed for the functioning of the location tracking, including the "uuid" and "major," which are both numbers you will get from your particular beacons by scanning them with a beacon info app. A name is also provided for each area, such as "Elevators" or "Stacks" to help identify that particular location. Finally, the configuration offers a testing variable, that help prevent false positives and can be adjusted based on how close you place the beacons and their relative output power.

BATTERIES

The Smart Commons Module uses two battery sources to get as much data collection done between charges as possible. The first source, which plugs directly into the JST plug on the C.H.I.P. is a 2,500 mAh lithium polymer battery from Adafruit and costs approximately $15. This battery can only sustain the module for roughly 5 hours, and it is the battery level that the system reads from when it writes data. The second source is a 20,000 mAh battery bank from Monoprice that uses a microusb connection to provide "external" power to the control-

FIGURE 2
C.H.I.P Controller and Battery without Module Attached



ler and costs approximately $20. It is only once this external battery bank dies that the smaller li-po battery begins draining, meaning that for the majority of the run time of the controller, the battery level will report as full. When the main battery bank dies and the C.H.I.P switches to the li-po battery, that is when the voltage drops and the system will start sending out alert emails to have them charged. A smaller battery bank or no battery bank could be used in order to save on the cost, but the system would require far more frequent charging.

PIEZO ELEMENT

The alarm system on the modules uses a Large Enclosed Piezo Element from Adafruit ($1) to create the noise when the alarm is tripped. The configuration file takes a frequency for the piezo, which dictates how high or low-pitched the sound is. In general, the piezo will function more efficiently at certain frequency, which can be found on the datasheet for the item. The configuration also has an option for the duration, which is currently just set to an absurdly high number so that it does not timeout if the alarm is tripped.

MISCELLANEOUS

The system also uses a few other miscellaneous pieces of equipment, such as standard jumper wires, buttons, and resistors, all of which cost a few cents each. The most notable of the other equipment is the ¼ perma-proto board that the module is soldered to, which costs approximately $2.50. Any proto board could be used for this project, but the perma-protos are easy to transfer from a breadboard testing setup and the ¼ size option is space-efficient enough to fit in the custom case designed for the project.

PIR SENSOR (EXPERIMENTAL)

This sensor is not part of the Smart Commons Module that is attached to the chairs and whiteboards, but instead is used in our group study rooms to detect when the space is available or in-use. Unlike the other controllers, this

FIGURE 3
Closeup of Smart Module

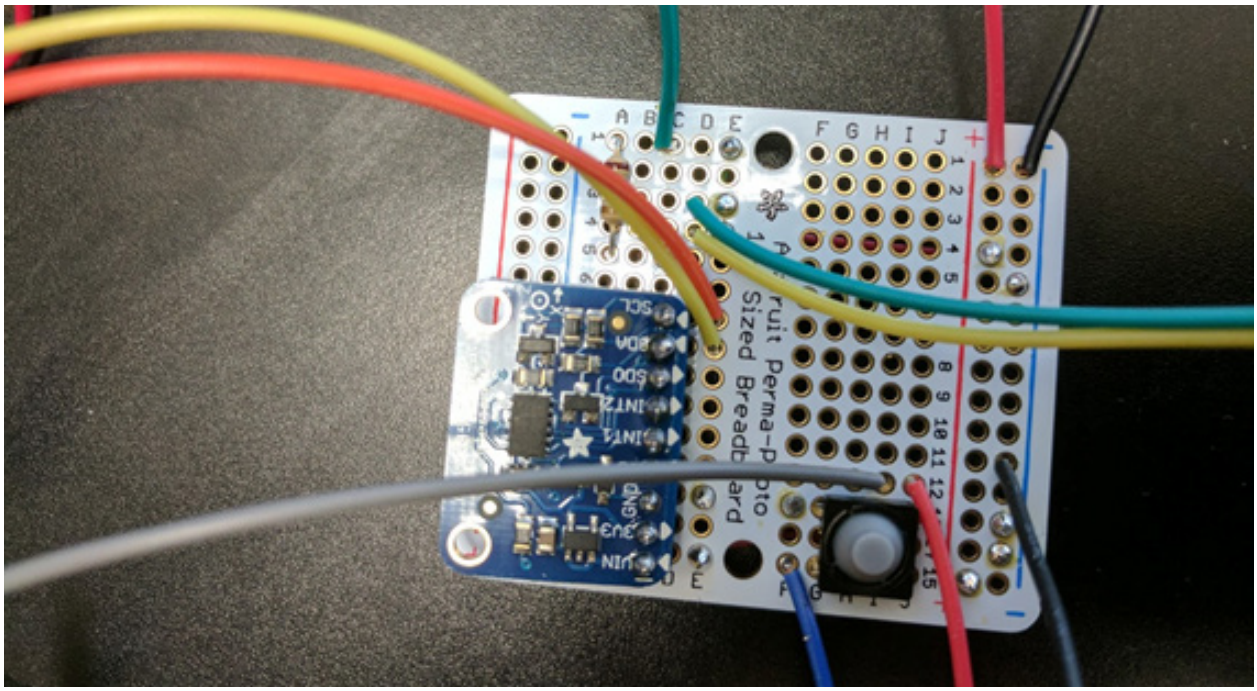
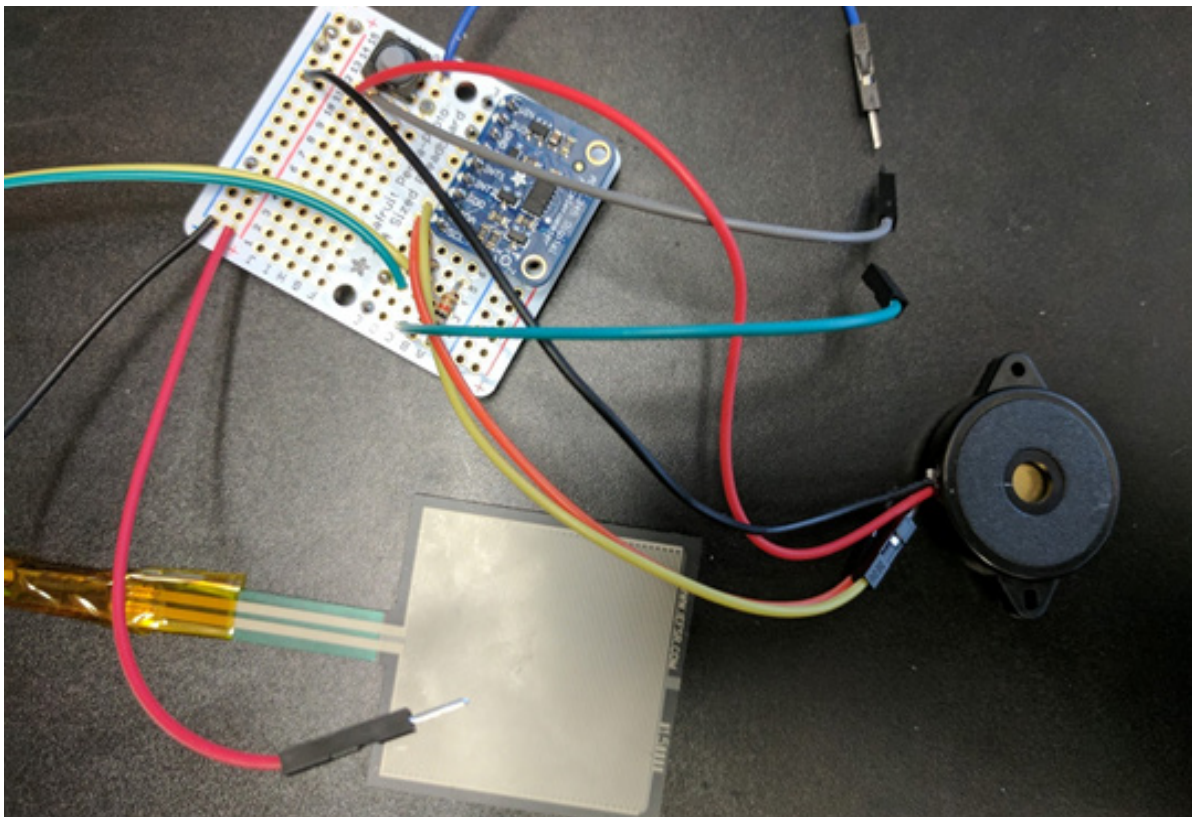


FIGURE 4
Smart Module with FSR Sensor and Piezo Element



one requires a hardline to power and will broadcast to a server instead of database so that students can check a feed of information before entering the building. The PIR Sensor we used came from Adafruit, and cost approximately $10.

THERMAL CAMERA (EXPERIMENTAL)

An alternate module is being developed that takes advantage of the FLiR Dev Kit Lepton Thermal Camera module from Sparkfun, which costs approximately $259. The camera is tied to a Raspberry Pi 3, but using it allows for more detailed analysis of space usage. Using the Open Computer Vision library, students can actually be tracked in the space as they move from zone to zone, with data collection happening actively. This module, like the PIR sensor, requires a hardline power connection, but this setup offers a more detailed solution to questions about group study behavior and also more common tasks like gate counting.

The Software

Node.js

The software runs a Node.js instance, a JavaScript environment that interacts directly with a server's underlying system, allowing it to tap into the hardware sensors directly. The system allows for easy configuration and installation for beginners, and could be extended to offer a server capable of streaming real-time data.

Johnny-Five

A Node.js package, Johnny-Five allows for the easy control and reading of sensors and other components for processing. The Johnny-Five code included in this project uses a special add-on package designed to specifically interface with the C.H.I.P. controller called chip-io. Like the other software used in this project, Johnny-Five is open source.

Miscellaneous

The project also uses a number of other common open source Node.js packages, such as 'request' and 'bleacon,' which handle the data calls out and the communication with the Bluetooth beacons respectively. These packages are included in the project folder, and they are installed and configured automatically when the project is initiated according to the documentation on the repository.

The Data

The data is collected to a MySQL database with a fairly standard REST API sitting on top of it. This database is held on a server local to our library that is maintained by our IT department, and the API it uses will only communicate with the specific devices we have named. The schema for the database is mostly composed of timestamps for events, including when a sensor is triggered and the duration of the triggered. For each event, location and battery voltage data is also collected in order to provide a better mapping of data in visualizations.

Security

The system incorporates a number of security measures to help prevent both theft and hacking. The custom case design incorporates two buttons that, if depressed, set off the alarm buzzer and send a message to the server, which then notifies library faculty via email that the device is being tampered with. The buttons are placed in such a way that if the lid is removed, or the device is detached from the bottom of the chair, the alarm will trigger.

FIGURE 5
Top View of Smart Commons Custom Case 3D Model

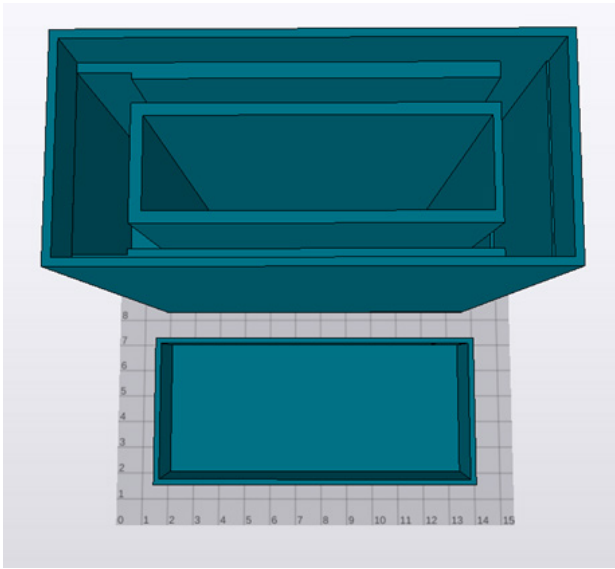
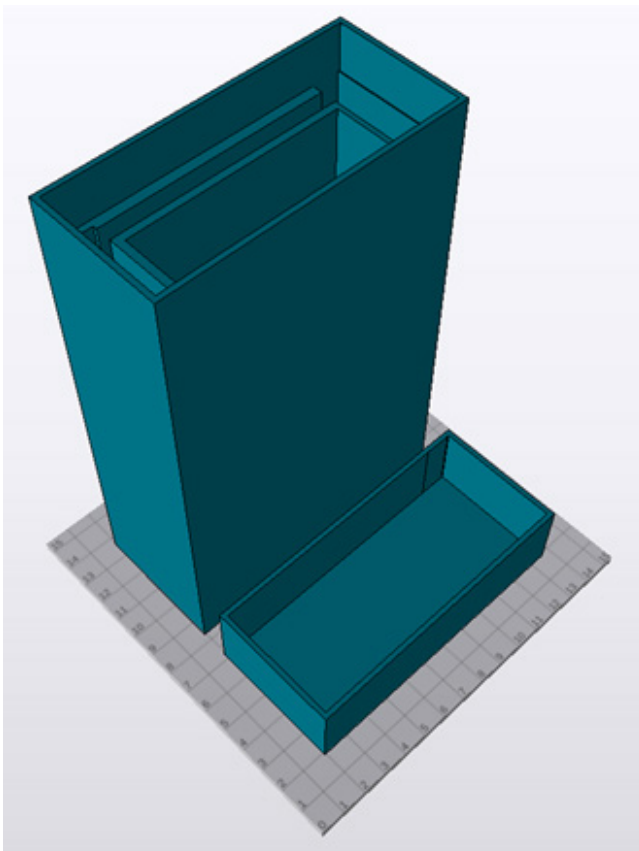


FIGURE 6
Front Corner View of Smart Commons Custom Case 3D Model



Hacking is deterred by the conscious decision to not make the C.H.I.Ps servers themselves or listen for input externally. Each of the controllers can call out to the data server, but they themselves don't act as servers, and gain all of their state information, such as whether or not the alarm is active or not, during their outgoing calls.

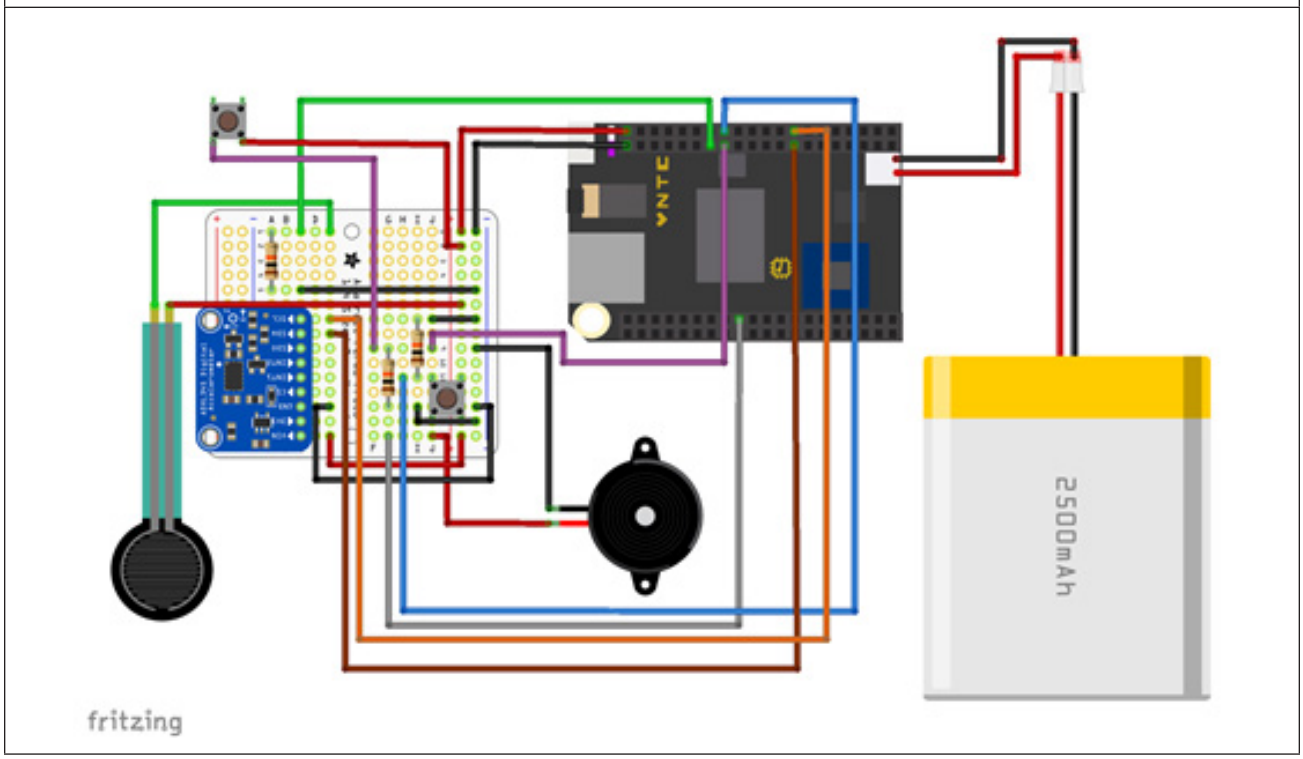
Documentation

Documentation has been provided in the repository, including a guide to preparing one of the C.H.I.P.s for use, a schematic containing the default wiring patterns and components used in this project, and code comments concerning the configuration of the software. There are also printing settings for the custom case and photos of different portions of the module as it is assembled to assist with the building process.

Implementation

Once the modules have been soldered together and placed in their respective cases, the devices are attached to the bottoms of various chairs and whiteboards in the learning commons in Newman Library. Each device communicates back to the server via the building's wireless internet access points, and whenever a triggering action takes places, data about the event, including timestamps, current battery level, and current location, is recorded to a central database. While each individual sensor is able to measure one aspect of the chair or whiteboard's use, the group of sensors together can provide insight into how users utilize the furniture. One module can reveal what times of the day certain chairs are being used, how often students are shifting or moving around, the duration of time that they stay, and whether or not furniture is relocated to different areas to fill a gap the students sees. The module tracks where it was located before a move, which area it was relocated to, and during what times of the day these moves take place. This data can provide insight into need, in that it might become clear that multiple chairs are relocated from Area A to Area B around the same time of the day. Correlate this data with class schedules, and

FIGURE 7
Wiring Diagram for Smart Commons Module



it becomes clear that students getting out of the evening classes are looking for more study space in Area B. If Area B is a quiet study space, the library can take actions to make sure ample seating is available in quiet study areas for students in the evenings.

Once the modules are attached to furniture and out in the commons, they are most self-contained. They report data as it is triggered, report back periodically about battery level and location, and are set to sent emergency notifications if tampered with. Periodically, the alarms must be disabled by changing a value in the database and then signaling an alarm state update, which currently happens by simply sitting in the chair for a moment, and then standing. During the alarm state check, the module will pull its most recent state from the database and react accordingly. At this point, the module can be removed from the furniture and taken back to private area to recharge the batteries. After recharging, the device can be reassembled, the alarm reset, and the module reattached to the furniture.

Future Considerations

Experimenting with the implementation of a Smart Commons called attention to a number of issues that will need to be more fully addressed in the future. While discussed frequently in the context of embeddable IoT technology, they are particularly germane to the library environment:

Privacy. Although no data transmitted from the sensors embedded in furniture and equipment contained identifying information about library patrons, utilizing such systems has the potential to spark concerns about privacy. And while signage was in place describing the nature of the project and the parameters of its data collection, some library users might have felt uncomfortable using furniture or objects equipped with such sensors and avoided them.

Data. Supporting the entire data life-cycle of smart objects requires a flexible database schema that could be added to down the road as a new sensor was added or a new data point teased out of legacy variables. The current database schema has columns structured with these names:

- id
- device_name
- current_location
- battery_level
- battery_timestamp
- time_sat_down
- time_stood_up
- duration_of_sit
- time_of_movement
- time_relocated
- type_of_relocation
- alarm_state

Each entry in the database will only contain part of the full schema. The alarm_state, id, device_name, current_location, and battery_level will be a part of every entry. The rest of the columns will only have data in them if a particular sensor is the one triggering the event. This strategy allows for the addition of more columns in the future if new sensors are added since the new columns would follow the same policy as this set; they would only contain values if that particular sensor reported data.

Since the data is sitting in a database behind a REST API, we have the option of opening this data to the public for consumption, allowing anyone to tap into a real-time stream of information being collected from the sensors. In addition, we are currently developing a dashboard that would display this data in real-time, relying on the socket.io package for Node.js to keep bi-directional connections to the API live and constantly updating a series of graphs on a webpage.

Power. Energy consumption and efficiency remained obstacles throughout our project. Controllers like the Raspberry Pi and C.H.I.P., while powerful, are far more power hungry than smaller simpler boards that most IoT devices use. Most of our initial attempts to extend battery life, including the addition of the battery bank to the design, have ultimately been stopgap measures in anticipation of better technology releases on the horizon. The ultimate goal would be to get the battery life extended to the point that the modules only have to be taken down once a month or so, allowing for very little downtime on data collection and less maintenance for those involved in the project. This goal should be achievable with a controller like the ESP32, which advertises a sleeping power draw of only ~2.5 microamps, compared to the C.H.I.P.s idle power draw of ~230 milliamps. Even if we take the ESP32's claims with a grain of salt, it is still sure to over magnitudes longer run times, and with some code that takes advantage of sleep mode on the controller, the goal of a month between recharges is within grasp and the possibility of going beyond that number also seems reasonable.

With this change in battery life, an external battery bank is no longer needed either, which would allow for an exceptionally smaller footprint for the module itself and an easier to design case for attachment to furniture. Another side effect will mean less down time for the module, offering a more consistent and reliable stream of real time data from the dashboard.

Security. As the number of smart objects connected to the Internet of Things grows, so does the potential for a corresponding "Internet of Harmful Things": devices affected by simple botnets and other malware created

by hackers to scan, disrupt and steal data from them. As our project proved, sensors can also be altered the old fashion way—through physical tampering by malicious (or simply curious) library users.

Conclusion

Despite these challenges, our project demonstrated that the real-time delivery of information made possible through IoT-enabled devices could easily complement established qualitative and quantitative approaches to measuring user activities and experiences. The data harvested through the IoT adds a new and flexible dimension to evidence-based assessment of library learning spaces. As new materials for sensing and signaling are invented, the presence of the IoT within homes, businesses, and public spaces will continue to grow. Sensors, actuators, and other IoT components will undoubtedly become smaller, faster, cheaper, and more intuitive to use. The capacity for libraries to harness the power of smart objects to both create and assess new services and spaces will likewise expand exponentially.

But how will this enable and enhance learning in the library? Like the smart refrigerator with which this paper began, a smart commons will not only provide data about its use, which is useful to the librarian; it will also be able to *anticipate* the needs of its users. By doing so, it will facilitate the production of knowledge and encourage connections between people and ideas. Imagine walking into a commons and getting recommendations on your phone about where to sit based on the similarity of the research others working there are conducting. Imagine chairs that alert you when friends or colleagues are nearby. Imagine a whiteboard that is able to push scholarly articles based on words, phrases, or diagrams written on it. A smart commons is a connected one. Its components are networked, coordinated, and conversant, not only with users but—and perhaps more importantly—with each other.

Notes

1. Rifkin, Jeremy. 2014. *The zero marginal cost society: the internet of things, the collaborative commons, and the eclipse of capitalism, 13*.
2. "Libraries and the Internet of Things," *NextSpace: The OCLC Newsletter,* http://library.oclc.org/cdm/ref/collection/p15003coll11/id/23. Accessed February 1, 2017.

Additional Resources

Please visit Github for code and technical specifications: https://github.com/VTUL/smart-commons.