Reviewer 1

This is an interesting paper, presenting an open source approach for a complete solution to the problem of monitoring indoor environmental parameters inside buildings. The authors discuss in detail their hardware and software implementation, with very interesting aspects presented in the paper.

Author response: We appreciate the positive and constructive feedback.

One issue that I have with the paper is its title: at first sight, when seeing the "building energy" part, I imagined that the open source aspect would extend to the provision of energy monitors as well. However, this is not the case, since the authors rely on other devices to get such data, while for the rest of the environmental parameters they present designs of their own. For this reason, I find it a bit misleading to include the term "building energy" part in the title. For the control part, the authors showcase a custom PCB with such capabilities.

Author response: We understand the perception that ‘energy’ is not a core component of the PCB based sensor package as is, outside of smart meter communications. We debated whether or not to include ‘energy’ based on this as well, but we believe it is appropriate to keep “building energy” in the title for a few reasons, including: 1) the smart meter capture is a key part of the platform, especially for residences where this technology is becoming ubiquitous; 2) by explicitly monitoring radiator temperatures or other temperatures such as HVAC air stream with the existing PCB based sensors, we do collect data on aspects directly related to or comparable to building energy use; and 3) the platform can easily be extended to measure plug loads and HVAC or other equipment/appliance draw (in fact, a current transducer easily connects to our sensor terminals; we have mentioned this explicitly in section 2.2 now and are working on addition to the github for such a sensor).

I also would like to see a larger focus on calibration: although it is a big issue in such applications, it is not mentioned much in the text. However, the sensors mentioned in the text are generally compatible with what is currently used by the community and of acceptable quality. Regarding the radios used, I think a good idea would be to add some comparison/argumentation with respect to other similar technologies like LoRa which is a valid choice for this type of applications as well. I would also like to see more details on the performance of the radio. E.g., how do these radios perform inside buildings but in different floors, or with different types of material (walls, etc.) between them? The authors state that Elemental has been deployed in several apartments etc. (line 451), but do not get any description to what those buildings like, distance between nodes, number of hops, and so on and so forth. This would enhance the paper in my opinion.

Author response: We appreciate the emphasis on calibration, as we too believe it is crucial. As the reviewer points out, the sensors used are generally compatible with what’s used by the community. Moreover, the selection of sensors was born out of work we did back in 2014-2016, which involved a lot of calibration against commercial or research grade equipment, and then selecting the most accurate sensors at the time. We have explicitly mentioned this in the text and
 referenced not only our 2016 OSBSS paper, but also the masters thesis (from co-author Akram Ali) which contains even more calibration data for these chosen sensors.

For the radio selection discussion, we have added a discussion on why we did not select LoRa on in section 2.2: “Another popular wireless transmission protocol, LoRa, was also initially considered, but the RFM69 FSK radios were ultimately chosen due in large part to their simplicity. To connect LoRa radios to the internet, a LoRaWAN gateway is required. Existing gateways can be used, but for applications where data is sensitive, deploying and maintaining a commercial gateway requires precise installation and configuration to be effective and is usually more expensive than their FSK counterparts. LoRa is also slower than FSK protocol due to reduced transmission bitrates. FSK radios typically have enough power and range for indoor applications, while LoRa radios are better suited for large-scale deployments like dense urban or forest settings where interval of transmission can be lower and longer range in the order of several miles is desired.”

For radio performance, we have added a summary of our knowledge on this as well in section 2.2: “The range of the selected RFM69 radios was initially tested vertically in a dense high-rise commercial building in downtown Chicago, with successful data transmission between two floors (1,400 m² each), and laterally between two academic buildings with brick, glass, and steel construction, with successful data transmission from the core of one building to the core of another ~150 meters away.” We also now discuss some of our experience with radio performance in section 4.3.

I would suggest to add the following 2 references, both good examples of open source hardware design related to the work presented in this paper, and which also tackle related issues a bit more in detail. The first work presented an open source approach that could be used in indoor and outdoor environments in smart cities and stresses the calibration and data quality issues, while the second one deals with energy monitoring aspects explicitly:

Guillem Camprodon, Óscar González, Víctor Barberán, Máximo Pérez, Viktor Smári, Miguel Ángel de Heras, Alejandro Bizzotto, Smart Citizen Kit and Station: An open environmental monitoring system for citizen participation and scientific experimentation, HardwareX, Volume 6, 2019.


Author response: These are excellent references and we have added them. We appreciate the recommendation.

The paper overall is well-written, with a clear structure and very few mistakes (e.g., on line 291, "any many supported"). The figures are useful and easily legible. Some minor mistakes with the references as well, e.g., in line 310 perhaps the github repository reference is wrong.
Author response: We fixed the github repository and appreciate the detailed exploration to find the error.

Moreover, from what I understand the GitHub repository of the project is being enhanced this week - some additional details/examples would be welcome in the documentation.

Author response: Correct! We are still working on improving documentation in parallel with this submission and will continue to do so.

Also, as a final comment, although I understand that this paper required a lot of work to come to completion, in some parts in the description of the software it is not always clear what the authors did provide over existing solutions. Maybe it would be a good idea to make their contributions a bit more clear.

Author response: We have attempted to clarify, first here, and also in the text. Essentially, the backend is an application suite that contains InfluxDB, VerneMQ, and Grafana, each of which are existing technologies. However, they need to be configured and properly coded to work. That’s what the middleware does. It ties every component together. Here’s a very rough sketch of a sample program flow:

1. Wireless sensor captures data and sends to gateway. (custom code, using existing radio)
2. Gateway receives data and sends to Raspberry Pi over USB.
3. The Raspberry Pi has a custom firmware written in Elixir that handles incoming USB messages from the wireless gateway (and other gateways, such as weather station, smart meter, etc.). This firmware sorts through the data, understands which device sent what, and keeps it all lined up properly to send it to a backend server, if it exists/configured. In the campus building deployment case, the "backend server" is a local machine in our lab.
4. The backend server has our application called "Brood" that handles incoming messages, stores to database, brings up Grafana for visualization, etc.
5. The Raspberry Pi sends data to the backend via MQTT protocol, which is a standard way of communication between internet connected devices and a server in the cloud (or local).
6. The backend has VerneMQ configured - this is an MQTT message broker that handles incoming stream of messages from devices (in this case, a Pi, or in larger buildings, say 3-4 Pis). VerneMQ sorts through all of this, makes data available for various other parts of the backend. The sorting and configuring of data are all our custom code.
7. The "middleware" essentially takes data from VerneMQ, sends it to other apps, like influxDB (a database), where it is stored. The middleware is essentially the "director software" that is the main core code that is custom written to deal with all data, handle duplicates, set up configuration of influxDB, Grafana, do anything custom, etc. The core of any backend software is the actual custom code written to handle everything, and not just the existing technologies alone. We can use any message broker (like Rabbit MQ, Mosquitto) any database (like mySQL, MongoDB), any data visualization (like Chronograph, or plan D3 modules on an HTML website), but whatever you use, they all need to be tied together somehow to work seamlessly, regardless of the incoming data. That's what the middleware does. We specifically chose VerneMQ, InfluxDB and Grafana due to reasons listed in Section 3.2 of the text.
8. "Brood" essentially is a blanket term given to this collection of components, including our middleware. It preconfigures all components as well, so no one needs to mess with influxdb
configuration (which requires some knowledge of how that works) or VerneMQ certificates, configuration, QOS, etc. Everything is handled by Brood.

9. Brood also has other smaller parts, like custom libraries/code to generate certificates, configure SSL, initialize and monitor docker containers, also generate Raspberry Pi firmware and allow OTA update of Pi firmware, modules for client discovery, configuration of third-party cloud service providers like AWS, etc. These are all custom code that tie everything together.

10. The reason why we include pre-existing technologies in the term “Brood” is because everything is configured in a specific way and dockerized in containers, so it's actually also a custom deployment of existing technologies that come together in a particular way to make this all work.

Finally, to summarize, the components of the "Elemental Platform" include a combination of sensors/control nodes + Pis + Brood.

We have attempted to clarify these distinctions, primarily in the abstract.

Overall, I enjoyed reading this work and I propose to accept the paper with minor revision, after the comments above have been addressed.

Author response: We appreciate the careful review and thoughtful comments, and we have addressed each comment in our revision. We look forward to its publication.