*Reviewer: 1   
Comments to the Author   
In general - while the aim of your system is admirable I cannot really see how you are helping the implementer put data into your system. When you say you are compatible across multiple systems - do you simply mean that the user can write a client that just reads the data off the system and inserts the data into your database with some SQL? I get the point but it's not really an innovation. If you have a REST, SOAP or socket library that you use then detail it or if you have future plans for an interface then I'd explain that. I assume your file parsing library just reads in a static format? On the data presentation - concentrate on the plotting library tie in with the back end rather than saying the user can get data out with SQL and SELECT clauses - if the users knows SQL they'll say 'it's just SQL' and if they don't they won't understand it. I'd really try to give more concrete 'meat' to the system and if you intend to do things put in a 'futures' section to describe where you are going.*

Dear reviewer

Thank you very much for the interesting and insightful comments. Please find answers to your questions below. The changed parts of the article are highlighted with yellow background.

*Specifics   
  
\* I assume you import file format is fixed - maybe give an example of this?*

We do not have one single import file format as such. The majority of the data is logged directly from the programs that acquire it, which we have written ourselves. For the case where that is not possible, we write a data parser specific for the format the data is saved in. We presently have parsers for around 4-5 different file formats ranging from temperature loggers to XPS data. These formats are the creation of the equipment manufacturers and therefore I do not think we should give examples of those. I have clarified this point at the end of section 2.

*\* Do you have a framework to allow the user to input data into your system?*

We do not have a framework for manually adding data as such, but that is by design. We have focused on making a foundation for the implementation of the automatically run data logging clients. It has always been a design goal of this system that all data should be logged to the database automatically and as soon as it is possible. Besides the advantage of having the data present on the presentation website as soon as possible, this is also a backup concern. In our experience, if you rely on people having to upload the data manually errors will occur and it will from time to time be forgotten. By not providing tools for manual access (other than by SQL), we are encouraging all users to implement automation.

*\* I assume you are using 'standard' Python and not Python 3K*

Yes, but purely out of availability and adoption consideration. At the time of starting the implementation of this system, the Python 2 series was still the default for server installations, which often are more conservative about new packages. As soon as Python 3 becomes the default Python for such server installations, we will port the code. I have specified the version in the article in section 2.

*\* Reference NumPy and SciPy in your biblio*

These references have been added.

*\* With the continuous measurement and a local queue to hold data (both good ideas) do you have a framework for this and/or do you plan one in the future?*

No, much as for the ordinary measurements we do not have a framework for it as such and this is by design, for the same reasons. We have focused on providing the design criteria for the data logging clients and supplying example implementations. I have added a link to where these example implementations can be found to the article at the end of section 3.

*\* You have a big body of Python code - maybe break it into pseudo code with comments to explain what you are doing*

This is an excellent point. I have added a section (section 5.2) to the data presentation section that explains the structure and design of the code, which should help readers, maneuver the code.

*\* When you say non-centralized systems - give an example even if it is just a case of Excel files floating around on desktops!*

Actually the meaning of “non-centralized systems”, in that sentence, is the kind of distributed data systems mentioned in the reference at the end of the sentence. But you do have an excellent point about mentioning examples of what we came from. I have clarified non-centralized and added examples of traditional files in section 4.

*\* You explain your iof MySQL but you could have equally used Postgres for that so why didn't you? If it is just a flip of a coin - that's cool!*

Yes, you have a good point about that the requirements of our system is such that they could equally well have been meet by plenty of other SQL-servers. We choose MySQL because it did satisfy the requirements and there was already experience with it within the group. I have added a clarification about the selection of database to section 4.

*\* You says that a MySQL instance needs aa high performance server - unless you are uploading massive quantitites of data then you don't. Teh same of storage capacity - unless you have massive BLOBs you won't need that much storage. If you focus on storage talk about RAIDs - in layman's language obvioulsy!*

You have an excellent point here. The point that we really wanted to make is not we require a high performance server, but a server with a high uptime, out of concern to the continuous measurements which are stored locally in memory until the server or network is back up, and out of backup concerns for ordinary measurements. I have clarified this in section 4. With concerns to disk space we use mirroring and LVM to allow for expansion. I will clarify and explain in the same section.

*\* When you say standard server software do you just mean a MySQL server?*

We meant a standard LAMP server installation. I have clarified this in section 4.

*\* Giving access to all the tables in a database is a \_really\_ bad idea - what about security; one client can affect the data for every other client.*

That is a good point. We do not actually do that, but I can see that we did not explain it properly. We do in fact have separate users for each experimental setup and while all users have read access to all tables, only the setup user has write access to its own tables. I have clarified this in section 4.

*\* ODBC is a bit dated; why not use a direct access and encrypt the username/password on the system. Also having a different username/password will require the DBA to give new passwords to every user.*

We use ODBC primarily because we still have a big body of Windows based Labview code. For Labview there are open Labview modules for ODBC, whereas for direct access you need to buy additional closed modules from National Instruments. That being said, we do actually use direct access in some of our Python based data logging clients, where this problem does not exist. I have clarified this in section 4.

*\* If every client needs to make a table that means that each client needs to make a table. If I have 300 clients they each need their own table? Why not have a client table to identify the data?*

Yes, having separate tables for each setup does impose a larger table admin burden. It does however have the advantage of having the data for each setup separate, whereby it is easier to evaluate size usage, to restore that setups data if it was polluted and it simplifies the SQL queries a bit for users that do data scripted data treatment where they pull the data directly from the server.

*\* You are using a weirdly formatted database; the metdata should be normalised out - why didn't you do this. There are valid reasons; look for NoSQL stuff but at the moment it just looks like a weird design.*

The reasons for this design are simplicity in the database design and in the queries to fetch the metadata. Since the metadata is stored in a per setup basis, the amount of different kinds of measurements that needs to be stored is small. This means that the penalty in empty fields, with metadata columns that are not relevant for any particular measurement likewise is low and therefore we decided that normalizing it out was not worth the extra complexity.

*\* On the data extarction - you're just describing SQL cut this section right down*

Yes. It was an attempt to meet experimentalist that do not know SQL half way. But you have a point that it is not suitable for this article. It has been reduced.

*\* You list possible DBMSs; include MS SQL - even if you don't liek it it is a very popular system!*

Agreed. This is an oversight. It has been corrected in section 4.1.

*\* The visualisation system is very interesting - write mreo about this and show examples of swapping out the plotters. Maybe talk about having setups which don't require files to be written on the server directly?*

Yes, that is a good idea. Figure 2 has been changed (as well as the caption) to also include the other plotting engine in an edited figure and I have added some more description about the system in section 5.

*\* Are you just using the plotting library and raw HTML/CSS on the browser client side - if you are using DOJO or jQuery add it in and reference it.*

We are using just HTML/CSS and a bit of Ajax (Java script).

Regards, on behalf of the authors

Kenneth Nielsen