Open Source Building Energy Management System

oBeMS - A Manifesto

The Problem We'd Like to Solve

In our energy reduction work with numerous clients, we see a high proportion of occasions, where a well managed building management system (BMS) or building energy management system (BEMS) could contribute to saving significant amounts of energy, but such projects are either precluded by initial capital cost, of fall into disuse because on-going revenue budgets are not available to maintain them.

Further, despite the high cost of energy monitoring / automatic meter reading equipment and services, and BMS systems, the flexibility of these is limited, possibly to defend the commercial interests of the manufacturers and installers.

The following comments apply in general terms to automatic meter reading (AMR), energy monitoring systems (EMS), building management systems (BMS), and building energy management systems (BEMS).

Generic problems identified in discussions with clients to date include the following.

- Some equipment is rented giving rise to very significant annual charges. (Ten or more thousand pounds per year in some cases.)

- Some commercial offers also charge software license and update fees. (Thousands of pounds per year for large sites.)

- Manufacturers can withdraw products, or support for products, with little notice, at their discretion.

- Interoperability between products (even from a single manufacturer) may be limited.

- The costs of inter and intra site data links can be significant.

- Data formats and transfer protocols are seldom open standards or publically available. This makes the deployment of the equipment outside of the manufactures intended scope of use difficult, and in some circumstances the necessary reverse engineering might breach contract or be illegal.

- Closed system architectures allow manufacturers and installers to charge high prices for equipment and data manipulations which are in essence straightforward.

- The constraint is frequently imposed, that data processing must be carried on the service providers servers. This leaves the client no alternative to

paying whatever the service provider wants to charge, and also leaves them, vulnerable if the provider ceases trading, or is taken over by another company.

What We Think Is Needed

To overcome the above issues, we would seek to find, and encourage the use of equipment and tools with the following attributes.

- Documented behaviour which conforms to open protocols and standards.
- Moderate cost.
- Ease of connection to other equipment using standard communication protocols and hardware, particularly IP over Ethernet.

This would bring many benefits.

- Flexibility. The design is in the hands of the end user.
- Lower development, installation and maintenance costs.
- Low cost intra and inter site networking of system components, ideally over existing corporate / domestic Ethernet infrastructure, using generic components.
- The archiving, aggregation, processing and analysis of data using any tool that the end user chooses or develops.

Unfortunately solutions with the above attributes are thin on the ground. Some excellent work has been done by http://openenergymonitor.org and http://www.re-innovation.co.uk/ and others, but the systems produced tend to be discrete sensors and data loggers rather than integrated systems.

The proposal then, is to combine sensing, data logging, user interface, decision making and control functions in a configurable environment which may be implemented on a single machine, or distributed around a network.

Implementation

Has this process started yet ?

- Yes. For our own use, we have developed a distributed system based on open code and protocols. All code is open source, and anybody who wants it can have it.

Does the project have a name ?

- As a working title we have called the project oBeMS, short for ‘open source Building energy Management System’, though much of the code might be reused in applications outside the scope of energy management and BMS systems.

What network architecture is used ?

- Server processes read or report on physical sensors. Client processes interrogate these with simple ASCII strings sent to numbered ports. The client can be located on the same machine as the server, accessing by port number at the loop-back address, or it can be accessed remotely by designating a port number and a static IP address anywhere on the Internet.

What software architecture is used ?

- Server processes are relatively dumb. They initialise hardware, pre-process data, and allow data access over the network. Client processes are relatively sophisticated, and store time series data from multiple sensors which can be manipulated over various time ranges to derive useful indicators. These indicators can be presented graphically to users, and used to make operational decisions to control physical hardware.

What can be sensed ?

- At the moment, tested server code is in place to read voltages and sense pulses on ‘no volt’ contacts.

- Analogue sensors either collect data when requested, or alternatively read data continuously, applying various averaging techniques to reduce noise.

- Pulse inputs generate interrupts which increment counters. The counter values can be interrogated by client processes.

- Client process sampled data at configurable intervals, and a collection of readings taken at the same time are referred to as a snapshot. Snapshots are linked to allow time series data to be collected, processed and acted on.

How is data represented ?

- Items of data have a long name typically used to aid human understanding. They also have a short name which can be used to identify data and specify where it should be used.

How is data identified ?

- Any data which can be identified by short name, including derived indicators, can be graphed, or otherwise logged or used to make control decisions.

What have we been using all this for ?

- T4 Sustainability Ltd has been using this software to monitor temperatures, as well as on site electricity consumption and generation, but any DC voltage monitoring or pulse counting function could be accommodated easily. Servers could be written to accommodate sensors with digital interfaces such as SPI and I^2C.

What hardware is supported ?

- Development began on the MBED controller, see https://mbed.org/handbook/mbed-NXP-LPC1768 but has now migrated to the Raspberry Pi Model B because of the richness of the Linux development environment and the hardware resource available, see http://www.raspberrypi.org/faqs

- Although the intent is to be able to run on very low power and low cost hardware, the client code has been compiled under Fedora on PCs using Intel CPUs.

The Future

What's the future direction of the project ?

- To a large degree this depends who gets involved with the project and what they want to achieve.

- The 'roadmap' from the point of view of the original developers is to make the system stable, and add features as required to meet their own needs, the needs of their clients, and the needs of the community.

What has to be done to make this an 'industrial strength' solution ?

- A number of features need to be added to maintain and improve stability and integrity.

- Stability (not crashing) should be maintained by investigating any tendency to crash, even after long periods.

- System longevity (the hardware lasting a long time) should be maintained by limiting the amount of data written to SD cards, as the number of times these cards may be written to without failure is a known weakness. This should be achieved by the use of ram-disk, network drives and remote storage where possible, as well as the use of very large SD cards where these must be used.

- The system might be booted from a read only file system to minimise the scope for corruption of, and lack of access to, the installed software.

- Launch the oBeMS processes automatically from rc or init scripts, such that they respawn on failure, and respawn in preference to forcing a restart of the system by the watchdog timer or by other means.

- A watchdog timer should be implemented to reboot the system in the event of unrecoverable whole or partial failure of the system.

- A user interface must be provided to allow easy configuration without recourse to recompiling the software. It would be an advantage to be able to reconfigure on the fly without needing to restart the process.

- While the existing oBeMS systems could grow to monitor large numbers of sensors and control significant numbers of actuators and loads, the design will need to remain scalable from small domestic single board projects, through to large and secure distributed corporate solutions.

- Errors should generate alerts, and be handled with as much grace as possible, without corrupting or losing data. Errors should be represented without compromising data.

What features are to be added to the system ?

- Sensible key box layouts.

- Sending of ‘self-documenting’ fields in response to pulse data queries.

- Configurable number of days to retain snapshot data.

- Graphing a configured number of hours or days data on each graph.

- Indication the status of switched hardware.

- Implementing server interfaces to control electrical outputs.
- Implementing user interface to control electrical outputs (HTML5 ?).
- Implementing a web socket interface to let users control the system.
- Implementing an external data store, probably accessed over the network, e.g. a database on a multitasking Unix derived system, probably PostgreSQL on Linux. The use of two or three such low cost machines could make the system very resilient, without a 'single point of failure', though in the first instance only a single storage machine need be used.
- It should be possible to interrogate the data store with SQL queries from any SQL capable data analysis package of their choosing, and the current graph code could be configured to run from the database rather than live snapshot data.
- Devise ways for end users to specify ways to manipulate and represent data on graphs etc.
- Vector graph output (.svg), as opposed to bitmap.
- Graphs with nicer fonts etc.

Who are the developers ?

- John Beardmore, managing director of T4 Sustainability Limited. Before working in the environmental sector, John worked as a software developer on amongst other things, market research analysis and presentation software, high availability resilient heterogeneous network systems to collect data for the UK national pop charts, and an Internet based simulation of the retail win market for training Deloitte's accounting staff. See http://www.t4sustainability.co.uk/?about/people/john-beardmore
- Sam Townsend, computer science graduate.
- Akhil Sudarsan, Nottingham University masters student placement.

What commercial relationships do the developers have ?

- As the managing director of T4, John will use this code as part of his commercial and non-commercial work. Others are free to do the same on the condition that all work and products developed from the oBeMS project are placed in the public domain on the same basis as the oBeMS project.

- This approach benefits the client as well as the community, as the client will have full access to the source code for all parts of the system, and the system can be supported and developed by the client or members of the community, as well as the original developers.

- In line with the policy statement on the T4 web site, “We spread information about sustainability issues and good practice at no cost, but charge a fair rate for time-consuming activities”. In other words, we are pleased to share what we have created for ourselves for the common good, but will generally charge if you wish us to implement systems or features on your behalf.

- Should oBeMS users decide that they wish to undertake the development or maintenance of the software themselves at any stage, or delegate it to others, there is nothing to prevent them from doing so.

- Third parties may also develop the software, and also extend the range and functionality and supported hardware. Such new features would be a benefit to the whole user community.

- By following an Open Source development process, some of the problems common to existing EMS, BMS and BEMS systems might be addressed.

What commercial relationships do the developers have ?

Although Open Source Software (OSS) has not been widely used in energy management, it has been the approach used in the following examples, which provide credible alternatives to their conventional commercial counterparts.

- The Linux Operating System.

- The Open Office / LibreOffice software suite.

- The OSS ‘Graphics Stack’, Gimp (image manipulation), Inkscape (vector drawing and art tool), and Scribus (high quality DTP).

OSS solutions are also increasingly found in embedded applications, for example

- the Android Smart Phone Operating System.

Even government is starting to notice what's going on, and support for Open Source software is starting to manifest itself as policy.

http://www.ogc.gov.uk/oss/OSS-policy.html used to sate, amongst other assertions, that:

> “UK Government will consider OSS solutions alongside

proprietary ones in IT procurements. Contracts will be awarded on a value for money basis".

See https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/61962/open_source.pdf for a more up to date position statement from the UK government.

It is our opinion that Open Source Software is now mature enough contribute to cost reductions in corporate building and energy management systems, and that contributing to the development of these tools will reap reciprocal benefits for the user and the developer community, by developing in cooperation rather than in competition with other parties.

It is also hoped that this project will make the purchase and maintenance of these technologies affordable to new communities of users.